1999-03

A Virtual Commanding Officer, Intelligent Tutor for the Underway Replenishment Ship-handling Virtual Environment Simulator

Tenney, Karl R.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A VIRTUAL COMMANDING OFFICER, INTELLIGENT TUTOR FOR THE UNDERWAY REPLENISHMENT SHIP-HANDLING VIRTUAL ENVIRONMENT SIMULATOR

by

Karl R. Tenney

March 1999

Thesis Advisor: Rudolph P. Darken
Second Reader: Robert McGhee

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 1999	Master's Thesis

4. TITLE AND SUBTITLE
A VIRTUAL COMMANDING OFFICER, INTELLIGENT TUTOR FOR THE UNDERWAY REPLENISHMENT SHIP-HANDLING VIRTUAL ENVIRONMENT SIMULATOR

5. FUNDING NUMBERS

6. AUTHOR(S)
Tenney, Karl R.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

While developing a Virtual Environment (VE) Ship-handling simulator for the Surface Warfare Officer School (SWOS) in Newport, RI, researchers at the Naval Air Warfare Center Training Systems Division (NAWCTSD) in Orlando, FL recognized the idea of integrating an Intelligent Tutoring System (ITS) to provide feedback to the student. The system, known as a Virtual Commanding Officer (VCO), would provide instructional feedback to the student to ensure that beneficial training occurs. The VCO would allow a student to conduct valuable training without a human operator present.

The approach taken was to survey cognitive architectures to find a notation to form a specification for feedback generated and delivered from a Commanding Officer to a ship-handler in training during an Underway Replenishment (UNREP). The cognitive architecture called SOAR was selected. A SOAR-like architecture was used to develop a VCO-ITS specification that resembles three Commanding Officer training method profiles. The profiles were then reviewed by qualified Surface Warfare Officers to validate their accuracy.

The result was a specification for a VCO-ITS with validated domain content in the form of three profiles. This specification was provided to NAWCTSD in support of their future efforts in the development of a VE Ship-handling simulator.

14. SUBJECT TERMS
Ship-handling, Virtual Reality, Intelligent Tutoring Systems, Interactive Learning Environment, Virtual Environment, Surface Warfare, Computer Simulation, Underway Replenishment, Computer Graphics.

15. NUMBER OF PAGES
117

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A VIRTUAL COMMANDING OFFICER, INTELLIGENT TUTOR FOR THE UNDERWAY REPLENISHMENT SHIP-HANDLING VIRTUAL ENVIRONMENT SIMULATOR

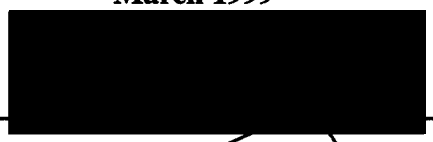
Karl R. Tenney
Lieutenant, United States Navy
B. S., Norwich University, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1999

Author: 

Karl R. Tenney

Approved by: 

Rudolph P. Darken, Thesis Advisor

Robert McGhee, Second Reader

Don Boger, Chairman
Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

While developing a Virtual Environment (VE) Ship-handling simulator for the Surface Warfare Officer School (SWOS) in Newport, RI, researchers at the Naval Air Warfare Center Training Systems Division (NAWCTSD) in Orlando, FL recognized the need to integrate an Intelligent Tutoring System (ITS) to provide feedback to the student. The system, known as a Virtual Commanding Officer (VCO), would provide instructional feedback to the student to ensure beneficial training occurs. Integration of the Virtual CO would allow a student to conduct valuable training without a human trainer present.

The approach taken was to survey cognitive architectures to find a notation that would form a specification for feedback generated and delivered from a Commanding Officer to a ship-handler in training during an Underway Replenishment (UNREP). The Cognitive Architecture called SOAR was selected. A SOAR-like architecture was used to develop a VCO-ITS specification that closely resembles three Commanding Officer training method profiles. The UNREP VCO profiles were then reviewed by qualified Surface Warfare Officers to validate its accuracy.

The result of this effort was a specification for a Virtual Commanding Officer Intelligent Tutoring System with validated domain content in the form of three profiles for an UNREP. This specification was provided to NAWCTSD in support of their future efforts in the development of a VE UNREP Ship-handling simulator.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

This research was possible through the assistance and support of many people and the sponsorship of the Office of Naval Research. I would like to recognize and thank Dr. Rudy Darken for his diligent guidance and support as my advisor. His absolute attentiveness and enthusiasm helped me carry this project to completion. In addition, I would like to thank the researchers of the Naval Air Warfare Center Training Systems Division in Orlando, FL for their technical support and assistance. My appreciation also goes to Professor Robert McGhee for his help on this project.

I owe a special thanks to my parents who have inspired me through the years. Their support and motivation has had a great influence on the successful completion of my goals while at the Naval Postgraduate School. I am truly grateful for the sacrifices they made to provide me the tools necessary to achieve this goal.

Finally, I am grateful to my wife Ann for her love and patience during my research and writing of this thesis. I truly appreciate her continuous support through my studies over the past two years.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

The specialized training of Naval Surface forces has seen numerous changes since the founding of the U.S. Naval Destroyer School in 1961. The school, now titled the Surface Warfare Officers School (SWOS), is continuing to train officers in the most efficient and effective way it can. Reductions in personnel and budget cutbacks have led the Surface Warfare Community to utilize new forms of technology to achieve its training mission. Training mariners at sea no longer remains the only option.

Increased demands to produce highly trained officers faster have led to the development of new training programs. The important aspect to any program is ensuring that quality training is achieved. Achieving quality training in this case means the teacher has a positive influence on the student, and the student walks away a better shiphandler. Creating an environment where students can practice, understand what happens when things go wrong, and do it again to get it right, is one solution [POOL98].

Though simulation has been an effective training asset for decades, the use of simulators to train Surface Warriors is still a relatively new concept. The Navy is investing in this technology to train its officers of tomorrow in a more timely and cost effective manner. Up to now, commercially operated, room sized bridge mockups requiring specialized operators have been the only simulators used to train ship-handling. The use of these simulators is in high demand, and due to their immobility, they remain an infrequent training option for many ships. Recent advances in computer hardware

performance along with declining prices have made commercial-off-the-shelf machines with enhanced 3D rendering capability more available. One such machine is the Silicon Graphics Incorporated (SGI) Octane™. Available for approximately $20,000, it is relatively portable and requires limited space to set up. As a result, designers of future ship-handling simulators can develop low cost and relatively portable systems that cross the surface warrior training spectrum.

B. OBJECTIVE

Training technology in the form of a high fidelity, real time, networked virtual environment (VE) ship-handling simulator would greatly improve the way officers are trained the skill of ship-handling. In order for modern training techniques to be accepted into the "old school" system, certain human performance and training requirements must be addressed and proven accurate. The specification for this system, though not concrete, is beginning to take shape in the form of an underway replenishment (UNREP) scenario. Eventually the system should include multiple scenarios, which can be modified to accommodate different levels of complexity for training at different levels of experience. Once in place, SWOS could utilize the simulator to teach the abstract concept of relative motion. In addition, SWOS could make accessible a training aid that officers could use to practice difficult evolutions in a safe, unbiased, stress free environment. Students can and should be afforded the opportunity to make mistakes, and to learn from them.

Researchers at the Naval Air Warfare Center Training Systems Division (NAWCTSD), with domain expertise guidance from SWOS and the Naval Postgraduate

School (NPS), are working on a project known as the Conning Officer Virtual Environment (COVE) system. Currently, the simulation system is focused on an underway replenishment (UNREP) scenario that is serving as a research test bed for future development. The UNREP evolution is one of the more complex and dangerous evolutions, yet it remains one of the most common evolutions for ships at sea.

Today, Junior Officers (JO) receive most of their practical training by observing and performing the evolution at sea. Since ships have been going to sea, domain knowledge has been passed from the senior experienced officers to the JO in the form of one-on-one guidance and feedback. For this reason, knowledge elicitation for the ship-handling UNREP scenario targeted experienced Surface Warfare Officers. After several months of domain knowledge acquisition, researchers have developed a high fidelity simulator that reasonably models reality. With continued refinement, researchers are using the broad knowledge database that SWOS and NPS students have to offer.

The next step in the developmental process of this simulator is to ensure that human performance and training requirements are met. The integration of an Intelligent Tutoring System that resembles the methods used by experienced senior officers to train junior officers would contribute to meeting these requirements. The objective of this thesis is to identify the common methods of training used by Commanding Officers (CO) and to create profiles based on those methods to support the further development of an UNREP VE ship-handling scenario. A secondary objective is to form a general specification in a cognitive architecture of a Virtual Commanding Officer (VCO) that could be modified to support future VE scenario development.

C. THESIS QUESTIONS

The following questions are addressed in this thesis:

- *What specific training methodologies are most commonly used by CO's during an UNREP evolution?*
- *How can an Intelligent Tutoring System be used to improve current training effectiveness?*
- *Which cognitive architecture is best suited for integration into a shiphandling simulator?*

D. APPROACH

This thesis conducted a review of teaching methods practiced by Commanding Officers and produced profiles based on those methods. Experienced Surface Warfare Officers were asked to review simulator-generated video segments of various runs and to comment on performance with corrective suggestions. These inputs were then evaluated and categorized into profiles. Next, the profiles were reviewed by senior qualified Surface Warfare Officers for validation. A general specification for a Virtual Commanding Officer (VCO) using the generated profiles was developed to support an underway replenishment scenario. To demonstrate an example of how an UNREP scenario might be coded, a general example of means-ends analysis in Allegro Common Lisp 4.2 was reviewed.

E. SUMMARY OF CHAPTERS

The remainder of this thesis is broken down into the following chapters:

- Chapter II provides a more in-depth view of ship-handling skill development, both in the schoolhouse and at sea. Additionally, the development and integration of intelligent tutoring systems into virtual environments is discussed. And finally, a brief discussion of Commanding Officer training methodology and the effects on training is provided.
- Chapter III discusses the fundamental components required to build a VCO and presents a methodology for the elicitation of domain knowledge.
- Chapter IV examines the VCO profiles created to replicate different training methodologies used by Commanding Officers during an UNREP evolution.
- Chapter V demonstrates examples of how the VCO could be applied to an UNREP scenario.
- Chapter VI presents a final discussion of the results of this thesis and describes areas requiring further research.

II. BACKGROUND

A. DEVELOPMENT OF SHIPHANDLING SKILL

The Surface Warfare Officers School (SWOS) in Newport, Rhode Island has started taking steps toward meeting the shiphandling training challenges of the future. SWOS is responsible for all Surface Warfare Officer (SWO) training, from the junior prospective Division Officer (DIVO) or Ensigns, to the senior prospective Commanding Officer (CO), Commanders and Captains. Providing effective training to students of this magnitude demands creative lesson planning and applied forms of practice. SWOS is looking at computer simulation as a significant part of the future of shiphandling skill training.

The SWO's of today acquire most of their shiphandling skill by training on-the-job. The following is a brief trace of the training SWO's receive during the early stages of their careers. Ensigns are sent as prospective SWO's to SWOS Division Officer Course (DOC) for six months to learn basic management, combat systems, engineering, and shiphandling skills. Upon graduation from SWOS DOC, junior officers are assigned to their first Division Officer tour on a ship in the fleet. Once there, they commence an eighteen to twenty-four month qualification process. This process includes completion of the required Personnel Qualification Standard (PQS) books, many hours spent conning the ship under the instruction of a qualified Officer of the Deck (OOD), and depending on the command, written testing may be administered. A junior officer must demonstrate good judgement and shiphandling skill, as well as gain the confidence of the Commanding Officer (CO)

before the CO will grant the qualification as an OOD. As a final step of the qualification process, the CO will convene a board of senior qualified OOD's to conduct an oral board exam of the junior officer's knowledge with regards to the duties and responsibilities of the OOD. In addition, shiphandling questions and scenarios are presented. By giving these questions during an oral board, the members are allowed to see how the junior officer would react in a given situation with little time to think about it. If the board is satisfied the junior officer has demonstrated the necessary knowledge, it makes its recommendation for qualification to the CO. After achieving this vital qualification, the junior officers hone their shiphandling expertise while standing watches as a qualified OOD. In addition, they will focus on completing their PQS books in engineering and combat systems to prepare for SWO qualification. After extensive preparation and passing a thorough oral board exam with the CO, the junior officer will be qualified a SWO. Generally, a junior officer will qualify SWO before moving on to their second Division Officer tour; approximately eighteen months on a different ship. Next, officers generally spend twenty-eight months on shore duty before returning to SWOS to attend the Department Head School. This school is designed to prepare officers to manage their respective departments, and refresh the skills that have become eroded while ashore, such as shiphandling. Graduates are sent to a ship in the fleet expected to possess the skill to properly train junior officers.

How are requirements of applied shiphandling training met? Up until 1993, SWOS used Yard Patrol (YP) boats to teach and practice shiphandling skill. These old wood hulled boats performed this duty for many years; however, their maintenance demands and

cost of operation proved to be too much for the budget cutbacks of the early 1990's. Shiphandling simulators were already being used and seemed a more cost-effective way to train. Today, the simulator available at SWOS is best suited for training navigation skills, not shiphandling skill. As a result, this simulator is only integrated into the Division Officer Course (DOC) curriculum. Since the simulator is unable to meet the demands of the advanced shiphandlers in the SWOS Department Head School, these students are sent to use a simulator located at the Marine Safety Institute (MSI) for approximately 16 hours out of the six month course. Thus, newly assigned Department Heads report to their ships with a limited shiphandling refresher, and are left to train and qualify junior officers with eroded shiphandling skills.

The MSI simulators are room sized bridge mock-ups that are viewed by many CO's to be effective shiphandling trainers. The bridge layout is effective for team training, and provides near realistic haptic and tactile feedback to each trainee. The simulators are limited to only a few locations for use by the Atlantic and Pacific Fleet ships. Since these simulators are in such high demand, SWOS is only able to provide its students with limited exposure to their training benefits. The simulators also require skilled technicians to operate, shiphandling experts to train, and are restricted by portability.

B. FUTURE SKILL DEVELOPMENT

The Navy is faced with the question of how to utilize technology to meet its training needs of today and the future. A look at current shiphandling simulator technology reveals two types in use or under development: the bridge mock-up and the

Virtual Environment (VE), which can immerse the user through use of a Head Mounted Display (HMD) or a desktop monitor. Though both are viable options, the development of a relatively low cost and potentially general purpose VE simulator that is portable, can be operated by the user, and doesn't require the presence of a human trainer, may be the answer the Navy has been looking for.

The bridge mock-up is an expensive option, requiring skilled technicians to operate while limited to providing training to a small number of students. In addition, a shiphandling expert must be present to provide feedback, otherwise a student's mistakes or questions would go uncorrected or unanswered. Usually the expert stands in the room to provide immediate feedback, and later provides a post-performance critique during a session debrief.

Mock-up simulators are the most widely used simulators applied to shiphandling training today. However, VE simulators are a less expensive and a more portable alternative. When experienced through HMD's, the student can become fully immersed into a VE. Since these systems can be designed to run on a desktop computer, the student can become the operator, and a human trainer can be replaced with an intelligent agent.

The Intelligent Tutoring System (ITS), a class of intelligent agent, is a very relevant concept when applied to VE shiphandling simulation. A new avenue becomes available specifically with respect to how and when the simulator could be used. For instance, simulators today require a human expert present to provide immediate feedback and critically rate student performance. Since these experts can only be present in one place and effectively evaluate one student at a time, an ITS with domain knowledge from

the expert could drastically expand training opportunities. Multiple students at many stations could be evaluated individually during overlapping periods of time. Also, students would be free to use the simulator at their convenience. This would promote confidence building without the pressure of performing in front of a known human expert. Given there isn't just one right way to perform a specific shiphandling task, a tutor must be flexible enough to support more than one method. In fact, there are usually a number of right ways to perform a task. With this in mind, the issue of tailoring the training method to match the criteria of human trainer becomes apparent. Simulators that utilize a human expert are limited to only one opinion. Is this enough? Hypothetically, an ITS integrated into a VE simulator could be tailored using an interface designed to pinpoint desired training methodology. For instance, an approach for UNREP can be done many different ways, all of which are correct. One approach takes a more direct path with small course changes while closing the range. Another approach may start by making the desired lateral separation before closing the range. A third approach might make a more defined course change while closing. All are acceptable methods to approach, which could be tailored into the ITS. The important point here is that the "correct" way to do an UNREP is how the CO *says* is the correct way. Junior Officers must be flexible in how they have learned to execute this task.

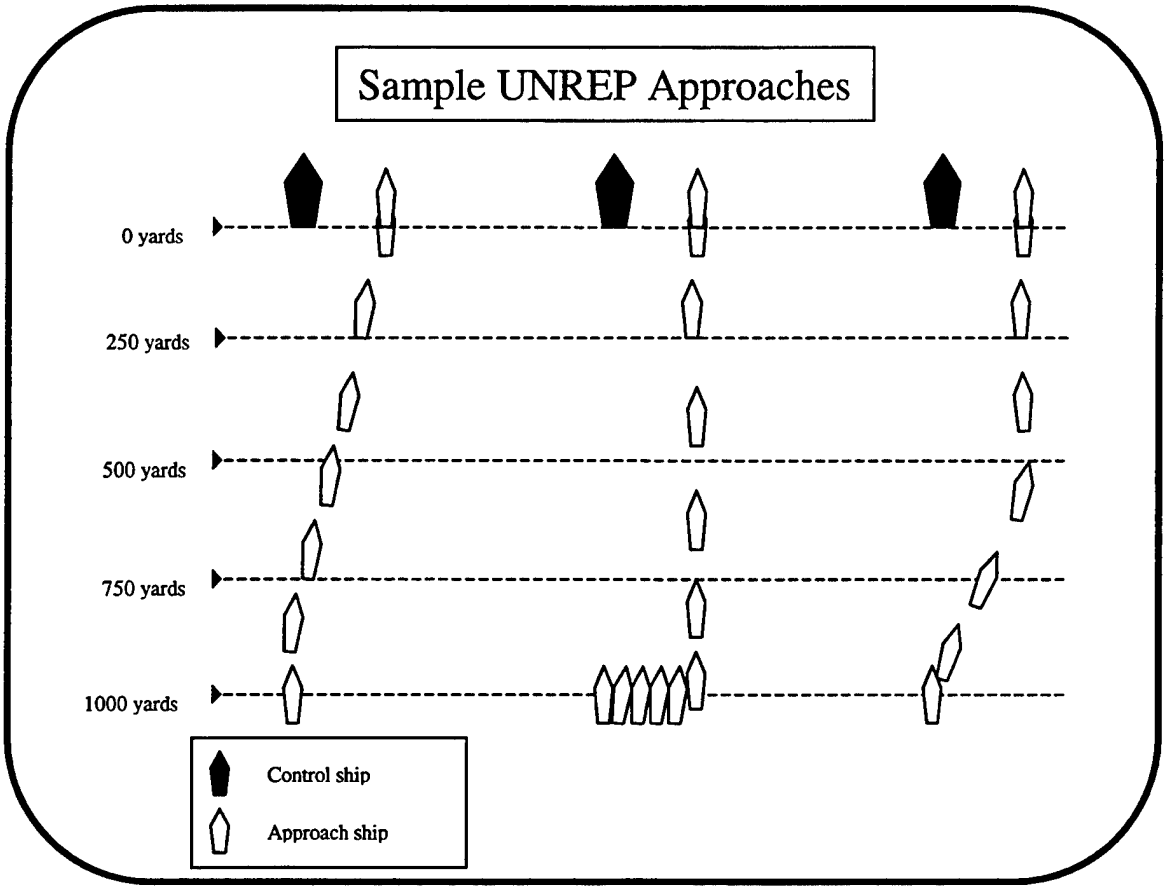


Figure 1: UNREP approach methodology

C. INTELLIGENT TUTORING SYSTEMS

In order to understand how a VE shiphandling simulator of the future can benefit from an integrated ITS, it is important to understand how these systems evolved, and what steps are required to develop one. To ensure effective training is conducted in Virtual Environments (VE), a system of guidance or direct feedback to the user is very important. One method of meeting this requirement is through the implementation of an intelligent tutoring system. The intelligent tutor can provide immediate guidance or feedback to the

user immersed in a virtual environment. When virtual environments are used in training simulators, it is imperative that the trainee receives guidance that is accurate or meets accepted standard. Failure to meet this requirement could result in a loss of valuable training time or a negative training experience.

Virtual environment simulators could be significantly better than mock-ups in that they enable a user, who usually wears a HMD, to be immersed into the environment. An important requirement for the successful use of VE's in some training applications is the use an intelligent tutoring system, sometimes referred to as an intelligent assistant or software agent. Development of these systems has evolved from the original expert systems, and continued interest has spurred advances in authoring tools designed to simplify the process. New virtual environments are currently being developed and implemented with intelligent tutors for use with various Department of Defense (DOD) and commercial training simulators. The Air Force Research Laboratory is funding the continuing development of Steve (Soar Training for Virtual Environments), an animated pedagogical agent that interacts with students in simulated environments. The goal of the activity is to create a usable instructional tool in a modular agent implementation that can be integrated into a variety of computer-based learning environments. [LAND99]

Computers have been used for educational purposes for almost 40 years. Since their introduction into the schoolhouse, there has been an increased demand for more advanced learning applications. Many of the original learning systems did not meet the high expectations that educators had set for this new academic medium because they were considered inflexible. These systems had limited capability for adaptive diagnosis and

feedback. [MAND88] Technological advances in hardware and software have made it possible to correct these shortfalls. Memory is cheaper and more plentiful, and programming languages are more powerful than ever. In fact, the developments in productivity enhanced programming has made it easier for psychologists and educators to participate directly in system design. Perhaps these advances laid the ground work for the continuing studies of ITS. [MAND88]

Intelligent Tutoring Systems (ITS) are software systems that can tutor people in a given area of expertise, such as engine maintenance, firefighting, or shiphandling. An ITS that is designed correctly models a student's understanding of a specific topic as they perform certain tasks. This performance is compared against a model of what the expert understands. If the two don't match, the system can use the expert model to generate an explanation to help the student understand where their performance could be improved. [LOCK98]

During initial development, one of the goals for tutoring systems was to create more powerful adaptive systems. To do this involves more than just examining a student or user's performance. In recent years, researchers have focussed on learning environments that are supportive and intend to facilitate learning-by-doing or transforming factual knowledge into experience. Intelligent tutoring systems attempt to combine the problem-solving experience and motivation of discovery learning with the guidance of tutorial interactions. [SLEE82] Many problems can occur between these two objectives. Since a user can drive a situation in any direction or instructional path, the system must have its own problem solving expertise, diagnostic or student modeling capabilities, and

the ability to explain the correct solution. In order to accomplish these capabilities, it must have explicit control or established tutorial strategy to decide when to interrupt a student's problem solving activity, to know what to say and how best to say it. All of this is required just to provide the student with instructionally effective advice. [SLEE82] Without some form of intelligent guidance, the student could struggle when faced with a new concept, or could move through a situation completely missing an opportunity that would result in high instructional impact given current state of experience or knowledge.

Another important issue involved in ITS is the development of a technology for expert system design and knowledge engineering. Early expert systems played a key role in helping educators see the potential of Computer Based Training (CBT). The name *expert system* comes from the term knowledge-based expert system. An expert system uses human knowledge and experience captured in a computer to solve complex problems that ordinarily require human expertise. Expertise is task-specific knowledge acquired from training, reading, and experience.

In the case of intelligent tutoring systems, the expert system's ability to solve complex problems is replaced with suggested solutions and advice. Expert system technology makes an attempt to transfer knowledge from experts and documented sources to the computer for the use of non-experts. Well-designed systems are able to imitate expert reasoning processes used in solving specific problems to a degree that non-experts can use the system as an aid to complete a task. This system would in turn improve their problem solving capability and help develop experience. In addition, some expert systems may be better than any single human expert in making decisions in specific, narrow areas

of expertise, usually referred to as a domain. These systems will make a decision based on all applicable criteria, where a human might forget or miss specific criteria. Development of early expert systems led to the following conclusions:

- General problem solvers are not strong enough to be used as a basis for building high-performance expert systems.
- Human problem solvers are good only if they operate in a very narrow domain.
- Expert systems must be updated constantly with new information.
- Understanding the complexity of a problem requires a great deal of knowledge about the problem area.

These conclusions serve as effective guidelines for the development of all intelligent systems.

Expert systems can provide many benefits. The most important are improvements in productivity, preservation of expertise, enhancing other systems, and providing training. A very influential aspect of expert systems on intelligent tutoring systems is their ability to provide advice in real time. The system's ability to react in real time has lead to its utilization as an intelligent assistant to the human expert and to the student user.

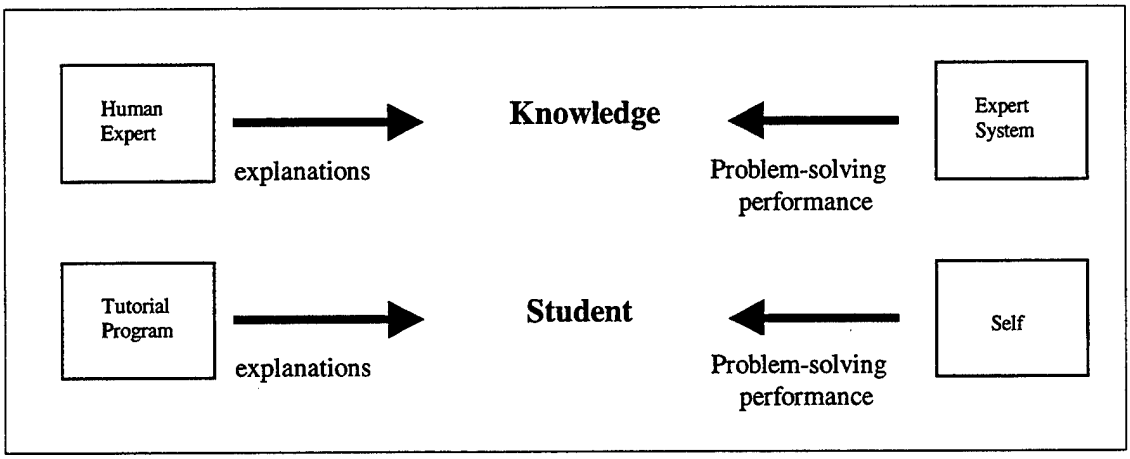


Figure 2: Learning analogy of a knowledge engineer and a student [MAND88]

The basis of an ITS is similar to that of an intelligent assistant. The intelligent assistant can be recognized as a system that supports effective usage of an application or simulation by monitoring the user's behavior and offering pertinent advice. An intelligent assistant consists of three major components:

- an articulate help base
- a user model
- a dialogue control module

An interface is used to interact with the user. The information is passed to the dialogue control module where the interactions between the user and the simulator are regulated. First the instructions are carefully selected. Then the tactful presentation, or "the how and when to present the information," is decided. The decision to present information is measured against the user model and the domain knowledge contained in the articulate help base. During all this, the user model is able to create and modify a profile of the

user's understanding of the current state of the simulation. This profile is compared with the articulate help base to identify differences. Diagnostic and presentation rules are then applied to the differences to generate output. [COXB98] A generic difference model used by tutoring systems is pictured below. [MAND88] A student's sub-optimal performance is explained as deviation from optimal performance.

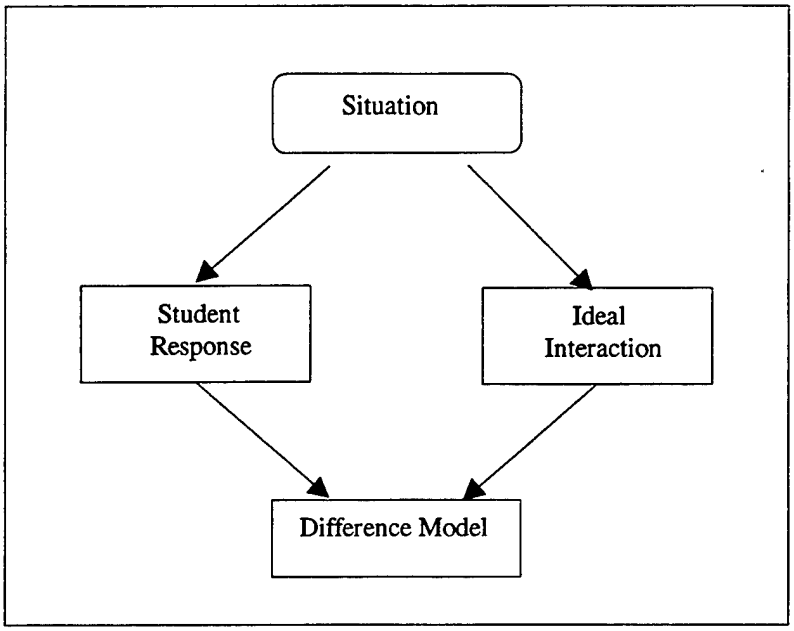


Figure 3: Difference Model

Development of an intelligent tutor starts with a pedagogical model. A pedagogical model is a computer program component that is build around a set of rules for teaching. Examples of such general pedagogical rules are:

- If the student makes an error, check the prerequisite knowledge.
- If the student is not responding, ask to see if he or she is confused.

- If the student is making a large number of errors and has been working for more than thirty minutes, suggest taking a break.
- If the student is doing very well, switch to more challenging material.

These are just some simple examples of teaching rules that aren't tied to specific content. [ALES91]

An example of an intelligent tutor under development at the University of Southern California Information Sciences Institute is Steve (Soar Training Expert for Virtual Environments). [RICK97] Soar is a cognitive architecture that could be used to generate an intelligent tutoring system for the VE shiphandling simulator. Steve, is designed to assist students in learning procedural tasks, such as the operation or repair of complex equipment while operating in a 3D, immersive, virtual environment. It provides various forms of assistance to the student to include task demonstration, student performance monitoring, and answering questions about which actions to take and the rationale behind those actions. Steve has been designed to support mixed-initiative interaction to ensure smooth collaboration between tutoring system and student when accomplishing tasks. [RICK97]

In order for a tutoring system to be intelligent, it must be able to react continuously to a user's learning pace, which may differ. Most tutoring systems utilize only one method of teaching. However, a human teacher often uses more than one method, specifically: lecturing, collaboration, and exploring. Teachers commonly switch from one method to another for material in the same domain to match different student

learning styles. Therefore, an intelligent tutoring system must be able to provide multiple teaching methods. [WARR98]

The building of intelligent tutoring systems seems to be based around one key objective. That objective is to create a system that can make inferences about student knowledge and can interact intelligently with students based on individual representations of what those students know. [MAND88] Included in this objective should be error correction and the utilization of a student model to decide when to intervene and teach the student how to perform a required subtask and when to allow the student to retain control.

What kind of structure does an ITS have? Typically the structure consists of four major components: the expert knowledge component, the learner modeling component, the tutorial planning component, and the communication component. [MAND88]

1. The *expert knowledge component*, as with expert systems, is comprised of the expert's knowledge of a particular domain. A critical part of expertise is the ability to construct an implicit representational understanding from observations and textual information. Old tutoring systems were limited to providing a calculated solution, but were unable to provide reasoning or an explanation to support the solution. Recent studies have focused on systems that can provide the explanations and more closely resemble human capability. [MAND88]

2. The *learner modeling component* is a dynamic representation of the knowledge and skill of the learner. As the learner performs certain tasks and interacts with the

system, a diagnostic deduces the learner's knowledge level. [MAND88] This deduction allows for the measured result against the expert.

3. The *tutorial planning component* regulates the interactions with the learner. It is the source and driver of pedagogical intervention. It is closely coupled with the learner-modeling component. It uses the knowledge of the learner and its own predefined tutorial goal structure to make decisions about instructional guidance, hints to overcome a performance stall, advice and support, explanations, etc. [MAND88]

4. The *communication component* is the final control of interactions between the system and the learner. It decides how to present the information to the user. Many systems are implementing graphical interfaces due to their ability to provide better concrete information output to the student. [MAND88]

The framework for intelligent tutoring systems resembles that of the original expert systems. The expert knowledge component, in its greatly improved state, and the ability to monitor interactions with a user to determine levels of knowledge and custom design feedback, provides for much greater instructional capability.

Below is a basic design of an ITS for use in a VE.

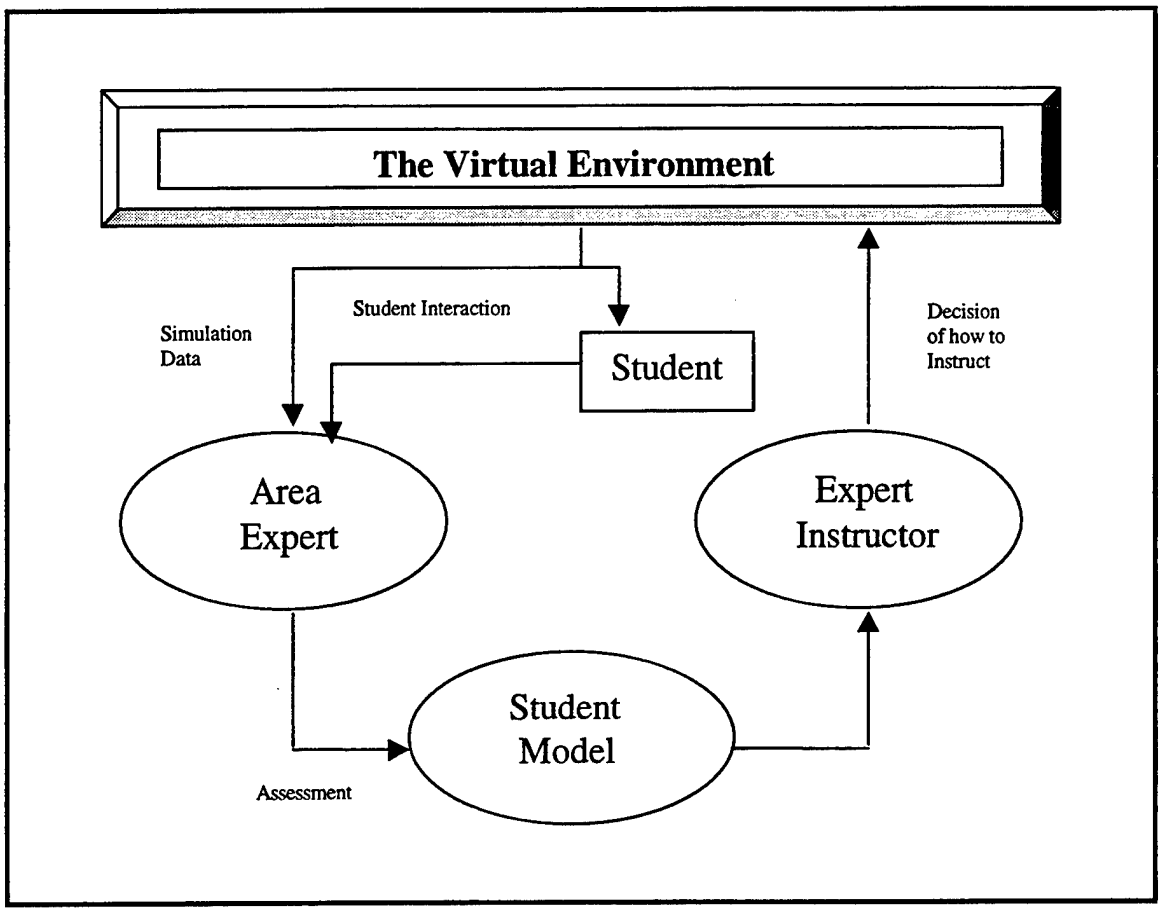


Figure 4: ITS design for Virtual Environments

The student and the local expert receive inputs from the VE simulation. The student's interactions in the VE are recorded and submitted to the local expert. The expert then creates an assessment of the student's performance to update the student model. The student model is provided to the expert instructor to decide how best to instruct the student based on level of knowledge and experience.

The issue of how to present instruction to a user in a VE is an interesting one. Since designers have gone to great lengths to provide realistic experiences in virtual environments, it is important not to disrupt the realism with distracting features. Most

tutors provide feedback in the form of text or graphical representation. In a VE, we must be careful how we provide input to the user. Text or graphical feedback could be more a distraction than benefit. Audible feedback seems like the best alternative. In real world interactions, teachers don't walk around with billboards telling students what they've done wrong and how to fix it. They correct them through demonstration and verbal interaction.

The future looks bright for ITS and its transition into VE shiphandling simulation. Real world training will never be replaced, but if designed right, the simulator can be used to increase the effectiveness of that training. Even if just used as a means of familiarization or task preparation, training in a VE simulator with an ITS attached could be highly effective.

III. APPROACH

A. ACCEPTED TRAINING METHODOLOGY

Every human trainer or Commanding Officer has a method for teaching the skill of shiphandling. Though one method may be preferred, they all work to achieve the same objective to train junior officers in accepted shiphandling skills. For the purposes of this thesis, these methods can be broken down into three major categories: aggressive method, passive method, and proactive method. This is not intended to be a complete list of methods but rather a representative list that has been agreed upon by SWO's surveyed for this thesis.

The aggressive method would be demonstrated by a Commanding Officer who takes control of a shiphandling situation to the extent of limiting the freedom a conning officer has to experiment with their own driving style. For someone just learning the basic shiphandling skills, this method has historically been desired. However, junior officers possessing the necessary skills to safely experiment do not typically prefer this method.

The passive method would be used by a Commanding Officer who intends to let the conning officer experiment with their skill. The CO only interjects to prevent dangerous situations from materializing. This method is better suited for training officers at a higher shiphandling experience level.

The CO who discusses strategies before the evolution and provides suggestive guidance before action is required would be classified as using the proactive method. This method is best for shiphandlers transitioning between low and high experience levels.

While there are preferred scenarios for each of these classifications of CO, they may appear anywhere at any time. For example, there is no guarantee a passive CO will not interact with a very timid junior officer.

Each human training method plays an important part in training shiphandlers. Therefore, this is a relevant issue for the development of an intelligent tutoring system. For the purposes of this thesis, characteristics of these three methods are transformed into three profiles for a Virtual Commanding Officer. It eventually might be preferable to work with the parameters that describe each of these profiles, but this is outside the scope of this thesis.

B. FUNDAMENTAL COMPONENTS

The ultimate goal for any intelligent tutoring system is the communication of knowledge. These systems include a special module, known as the *expert* that contains a representation of the knowledge to be communicated. Usually, this subject matter representation is not only a description of the various skills and concepts that the student is to acquire, as in a curriculum, but an actual model, which can perform in the domain and thus provide the system with the necessary dynamic form of expertise. In other words, a system has knowledge to serve as a reference, and is dynamic enough to adapt to the user's needs. [WENG87] An expert system can be represented as a knowledge base and

an interpreter for application to particular problems. Figure 4 depicts the expert system as one of two main components that form an ITS.

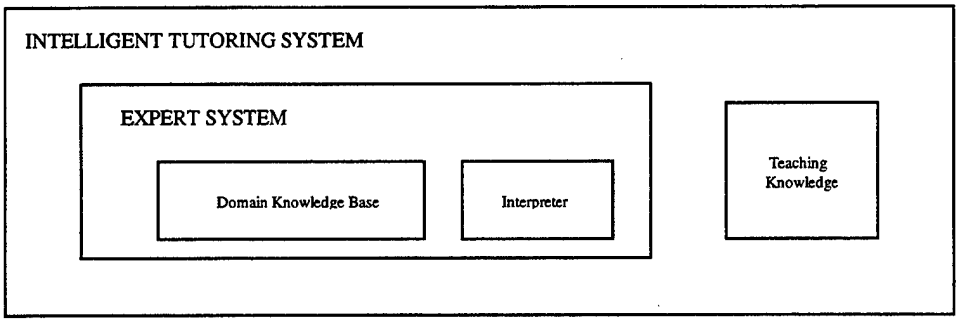


Figure 5: Components of an Intelligent Tutoring System
[KEAR87]

Though design is an important part, the intent of this thesis is not to uncover a novel form of building intelligent tutoring systems. The goal is to achieve domain knowledge elicitation, and define a strategy for inserting it into a tutoring system that will meet the criteria of an UNREP shiphandling evolution. In order to accomplish this goal, six sub-goals will be achieved: identify the domain expert, characterize expertise, validate the resulting model, determine the student model, choose the method of feedback, and finally, form a computational model.

1. The first step is to identify the domain expert. This is the human knowledge source that will form the domain knowledge base. The task of identifying the expert seems simple; however, it is impossible to pinpoint one shiphander as the expert. In this case, the expert is an experienced shiphandler with an established method of teaching.

2. The objective is to generate a model that resembles the expertise of an experienced shiphandler. What defines expertise? For shiphandling, can experience be considered the sole contributor to expertise? In some cases, recent experience may contribute more. An experienced Junior Officer may have just as much expertise, but less experience to back it up. Perhaps this answers why some relatively junior shiphandlers perform better than senior shiphandlers in the simulated environment.

3. A high degree of accuracy in any tutoring system is absolutely imperative. Inaccuracies lead to training bad habits, and bad habits can result in accidents. A step toward effective training can be assured by verifying the knowledge gathered. Review by senior experienced subjects is one way to accomplish this task.

4. A system tailored for one student is fairly simple to create, but only one student is not realistic for training applications. Reality provides students from various levels of experience and background. Students have different needs and attention requirements to train, so this factor must be accounted for when creating a tutor.

5. The choice of which form of feedback to provide depends on the environment that is being simulated. For instance, text feedback may be expected when the situation being simulated involves a keyboard and text display. However, for VE simulation, text feedback may not be the best choice. Perhaps audible feedback or use of an animated agent might be the better choice.

6. To demonstrate the tutoring system, a computational model must be formed. An architecture must be selected and a specification written. Once a design is complete, a coded model can be implemented.

C. METHODOLOGY

The task of knowledge elicitation was accomplished in three steps. The first step consisted of video review of selected segments associated with six decision points during an UNREP evolution. These six points were defined in the three phases of an UNREP: approach, alongside, and breakaway. The first three points were in the approach phase, defined as distant, middle, and close. The next two points were in the alongside phase and were defined as alongside and pre-breakaway. The last point was defined as breakaway. Each point segment, approximately 20 seconds in length, was viewed without sound inputs. Viewers were asked to comment on each segment individually, giving guidance that represents an aggressive training methodology. During the second step, the viewers were asked to make comments that resembled strategy planning. This step was meant to represent a proactive training methodology. The third step included review of the same video segments; however, during this review sound was included and comments were recorded. Viewers during this step were asked to make their comments in a way that would be representative of a passive training method. The comments were compiled and formed into profiles of the aggressive, proactive, and passive VCO. The profiles consist of an UNREP track with an associated CO training methodology, and recommendations at each of the six evaluation points.

The important step of verifying the accuracy of the knowledge gathered was completed by asking senior experienced shiphandlers to review the resulting profiles that

show the UNREP track with course and speed recommendations. Validator comments were recorded and reviewed to determine an outcome.

Who will be the student? VE simulation should continue to remain flexible in answering this question. Right now, the student ranges from the most junior to the most senior shiphandler. For the purposes of developing a tutoring system, the student that seeks to benefit the most from a VE simulator is a junior shiphandler. A senior, more experienced shiphandler, has probably already been exposed to different training methodologies and established their own style. For this reason, the tutor focuses on training the junior officer. The modeling of other students is an area open for further research.

The use of text feedback does not make sense in this scenario. In a VE, text is a distraction of visual clutter, potentially blocking visual cues and affecting the decision making process. Audible feedback is most realistic for this task, and was the mechanism selected. Use of an animated agent could allow the VCO to point out certain visual cues. This is something to consider for future research, but for now, audible feedback will be sufficient.

Finally, a Soar-like specification was written to cover the three phases of UNREP. Since Soar is easily compatible with Lisp, Lisp was selected as the coding language for a sample computational model. The model took on the classic form of a deterministic means ends analysis of a general machine assembly example.

IV. PROFILES OF THE VIRTUAL COMMANDING OFFICER

A. TRAINING UNDERWAY REPLENISHMENT

The number of correct ways to perform an UNREP is unknown. Each CO develops a style and expects certain key events to occur during the approach, while alongside and during the breakaway. Some may expect an approach speed of Flank 1, while others may be more comfortable with 5 knots over the control ship's speed. A CO may want to be steady on the replenishment course with 120 feet lateral separation before closing within 500 yards range. While training these styles into their junior officers, CO's take on a training methodology profile. Sometimes they may take on a combination of different profiles.

This thesis identifies three individual profiles that stand as examples of how a CO might train a junior officer. These profiles have been titled Aggressive, Proactive, and Passive. The process of creating the profiles consisted of three phases: During the first phase, five subjects, each qualified Surface Warfare Officers at the post-Division Officer level, were asked to view video segments taken from the COVE UNREP simulator. Four separate UNREP runs were evaluated. Each run was broken into six, 20 second, segments at prescribed points during each run. These points were identified as: APPROACH DISTANT, APPROACH MIDDLE, APPROACH CLOSE, ALONGSIDE, PRE-BREAKAWAY, and BREAKAWAY. Each subject was asked to make recommendations on whether to change course left or right, and increase or decrease speed. This phase did not include audio to encourage unbiased responses from the

subject. The comments were recorded and compiled to form the aggressive profile. During the second phase, each subject was asked to plan and comment on how they expect an UNREP to be performed. These comments were recorded and compiled to form the proactive profile. The third phase was a review of the same video segments from phase one, this time with audio. The repeat backs from the helmsman prompted fewer comments from the video review subjects. These comments were recorded and compiled to form the passive profile. The worksheets and additional scenario data provided to subjects is included in Appendix A.

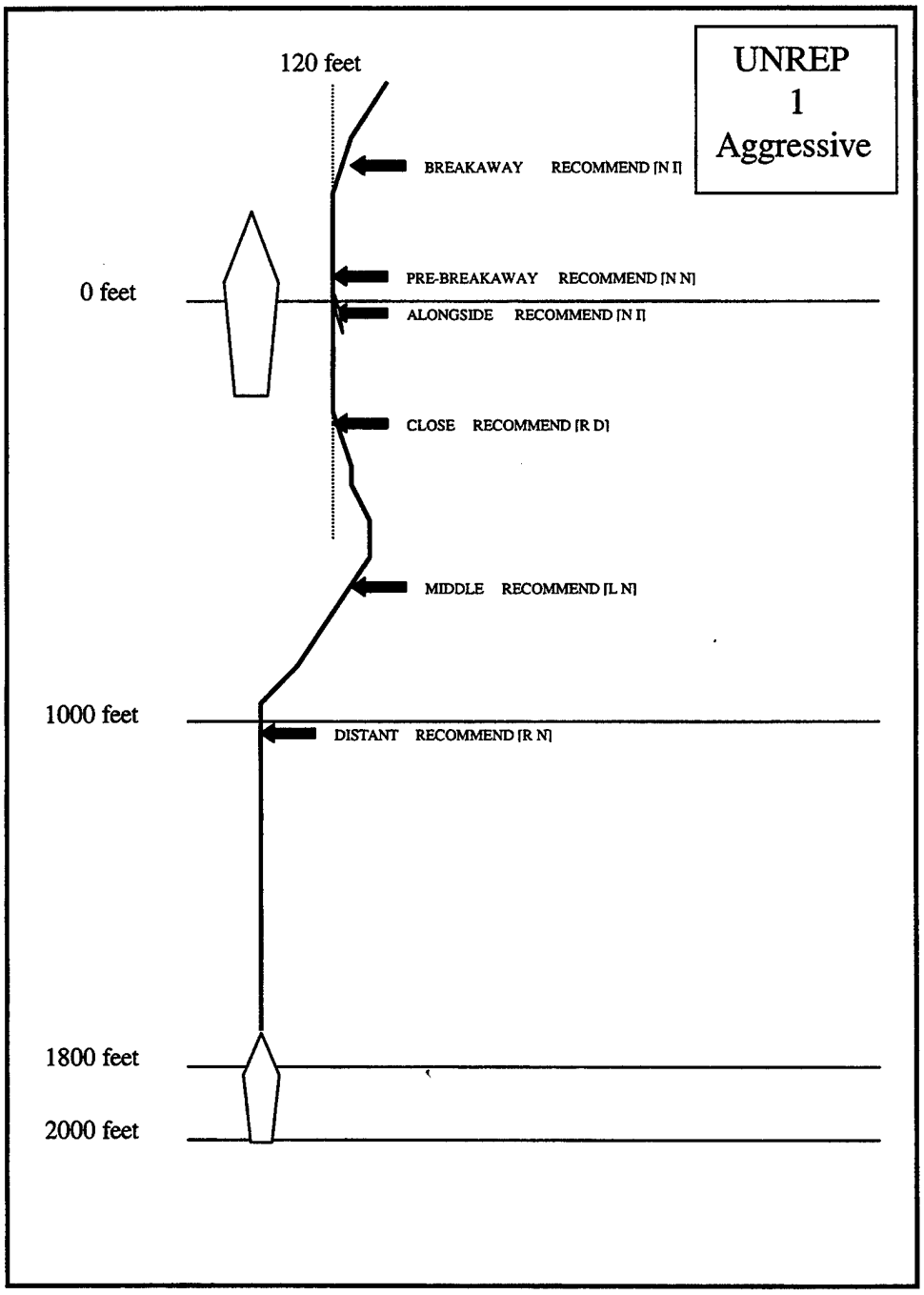
Track data for the four UNREP runs were generated, and the prescribed segments were marked. Then, the three profiles were applied to the tracks as recommendations from the CO to the shiphandler. The recommendations are as follows:

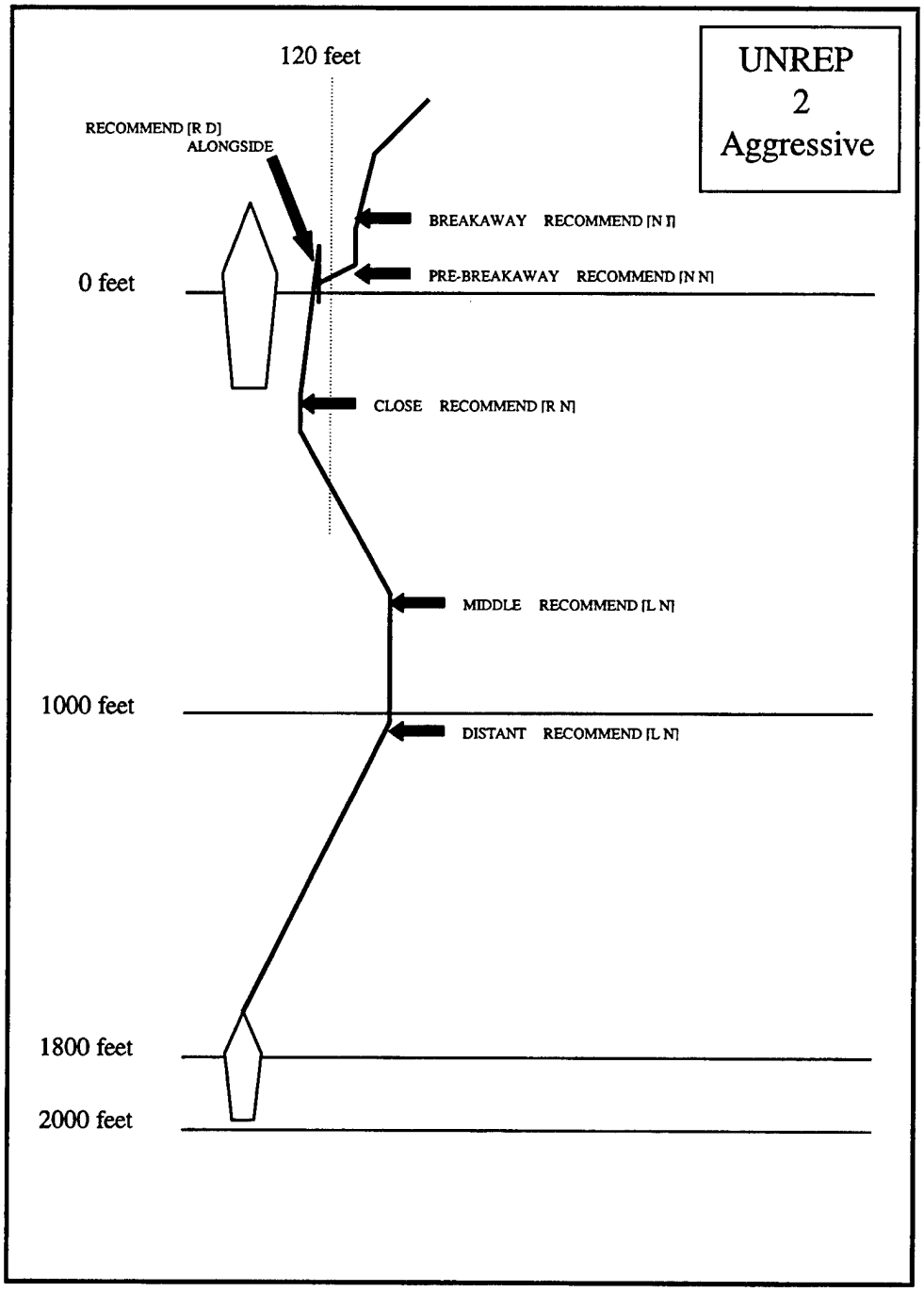
RECOMMEND [COURSE SPEED]

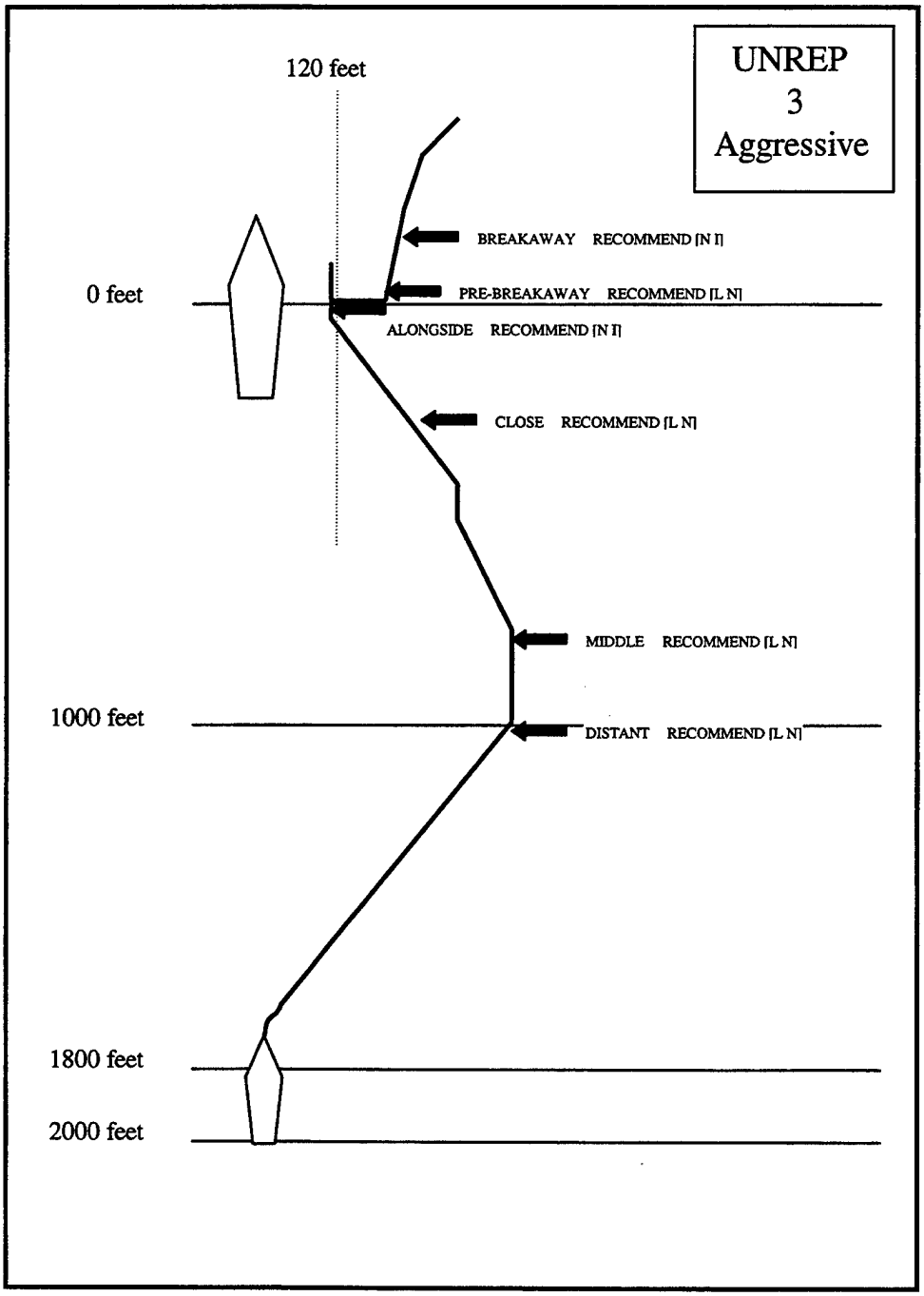
COURSE	L – Left	R – Right	N – No change
SPEED	I – Increase	D – Decrease	N – No change

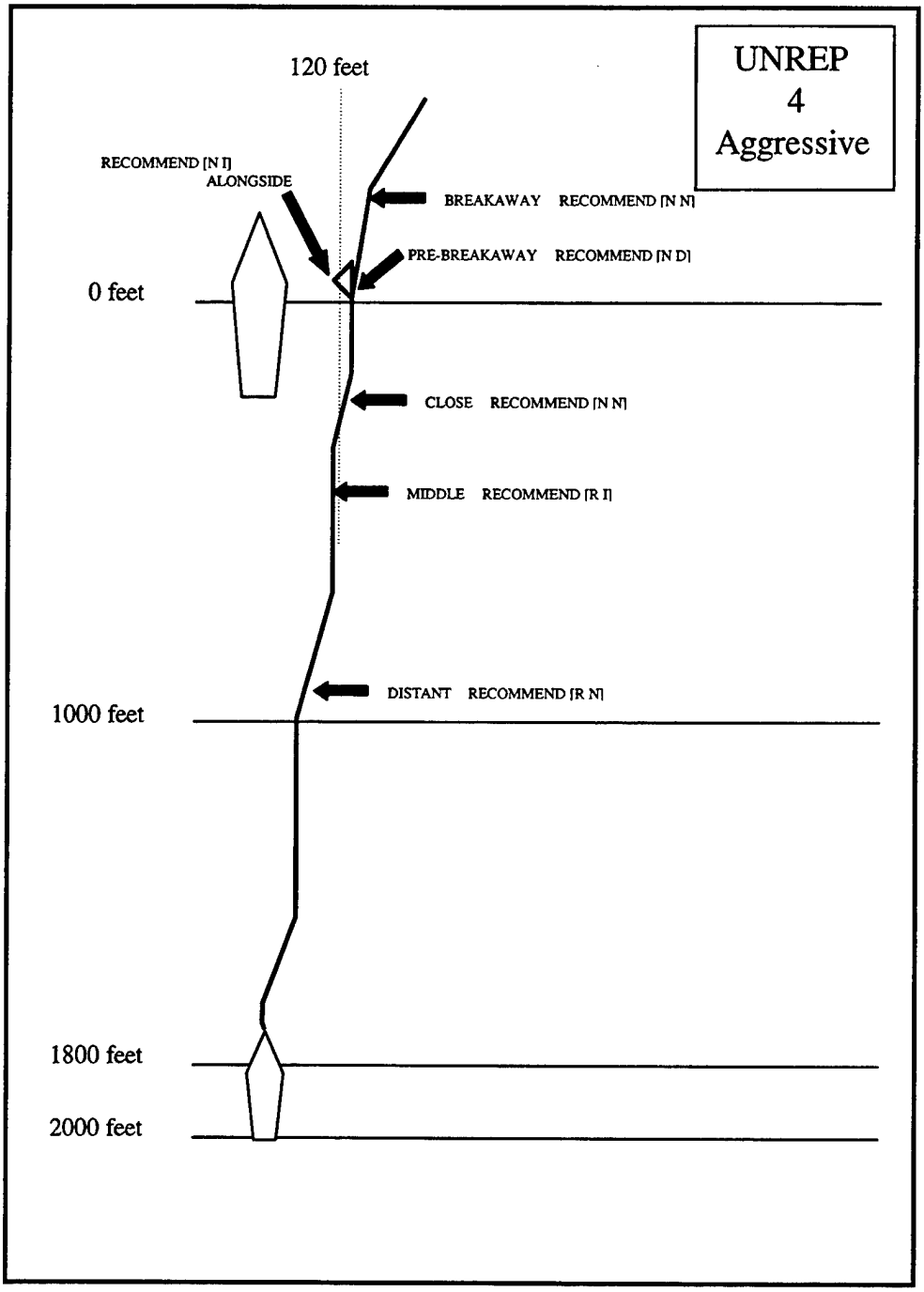
The following profiles are categorized by the three training methodologies. Each category includes a brief summary of the profile that identifies the characteristics of the respective CO.

B. THE AGGRESSIVE TRAINING PROFILE



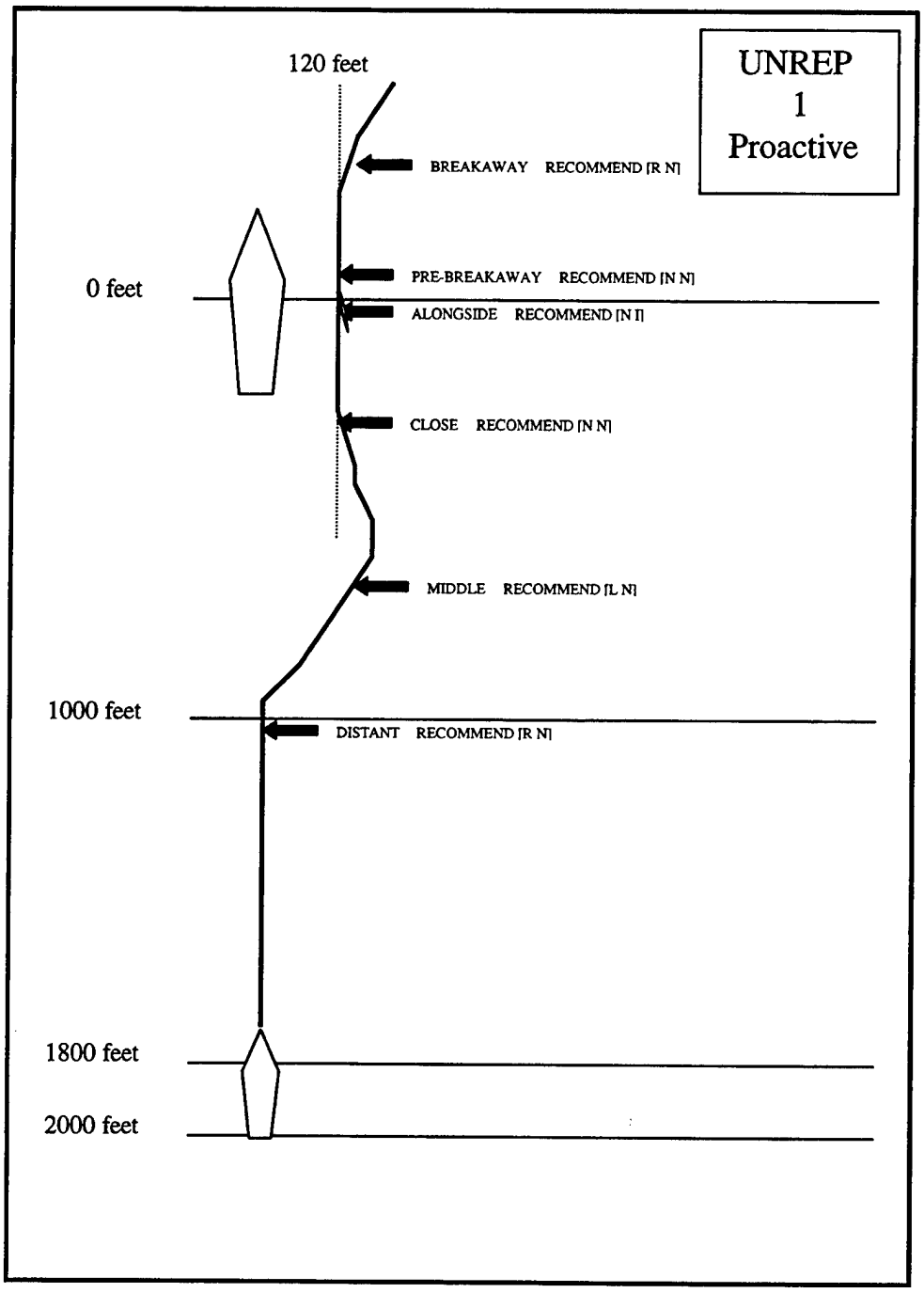

UNREP
2
Aggressive
120 feet
RECOMMEND [R D]
ALONGSIDE
BREAKAWAY RECOMMEND [N I]
PRE-BREAKAWAY RECOMMEND [N N]
0 feet
CLOSE RECOMMEND [R N]
MIDDLE RECOMMEND [L N]
1000 feet
DISTANT RECOMMEND [L N]
1800 feet
2000 feet

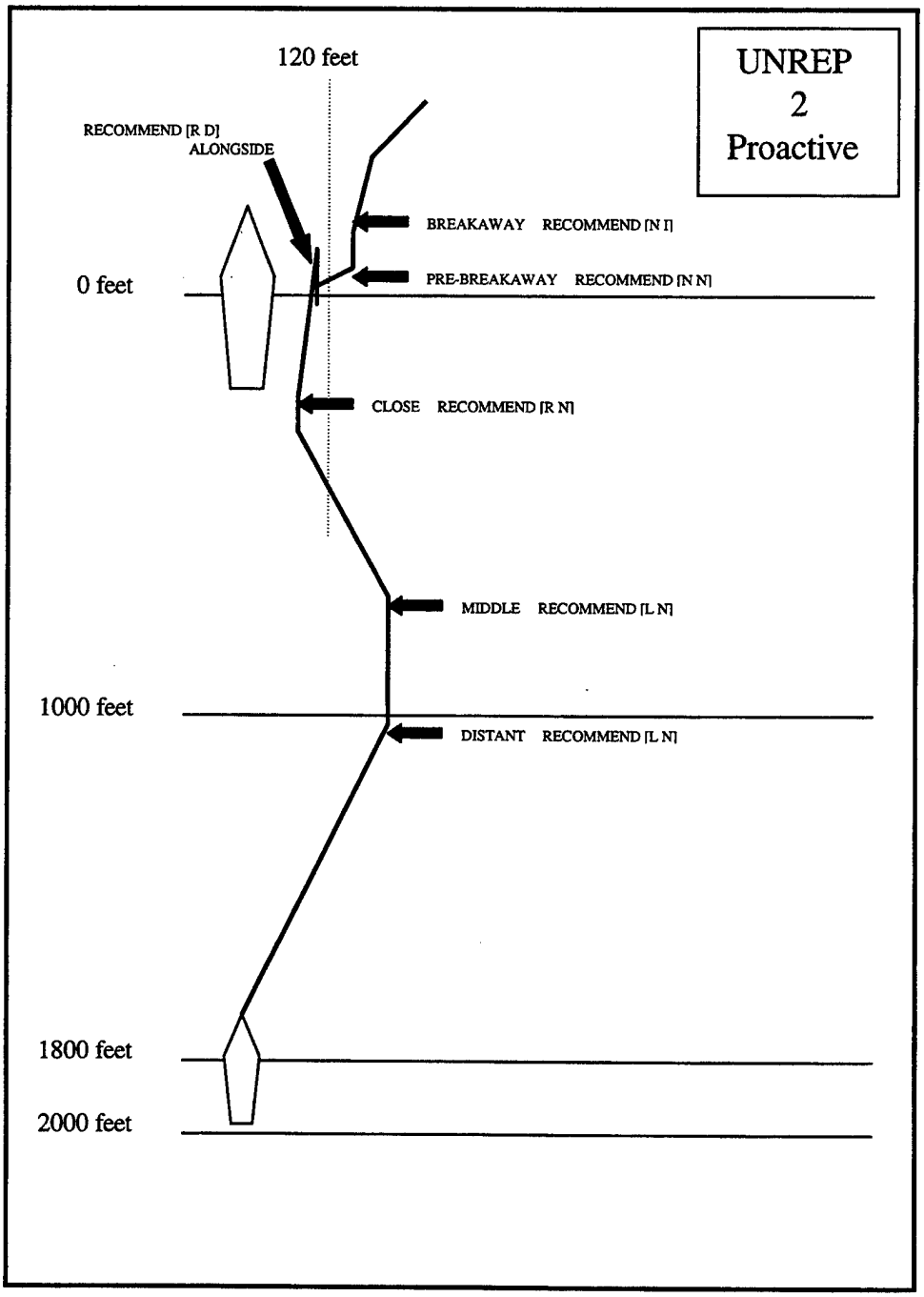

UNREP
3
Aggressive
120 feet
0 feet
BREAKAWAY RECOMMEND [N I]
PRE-BREAKAWAY RECOMMEND [L N]
ALONGSIDE RECOMMEND [N I]
CLOSE RECOMMEND [L N]
MIDDLE RECOMMEND [L N]
1000 feet
DISTANT RECOMMEND [L N]
1800 feet
2000 feet

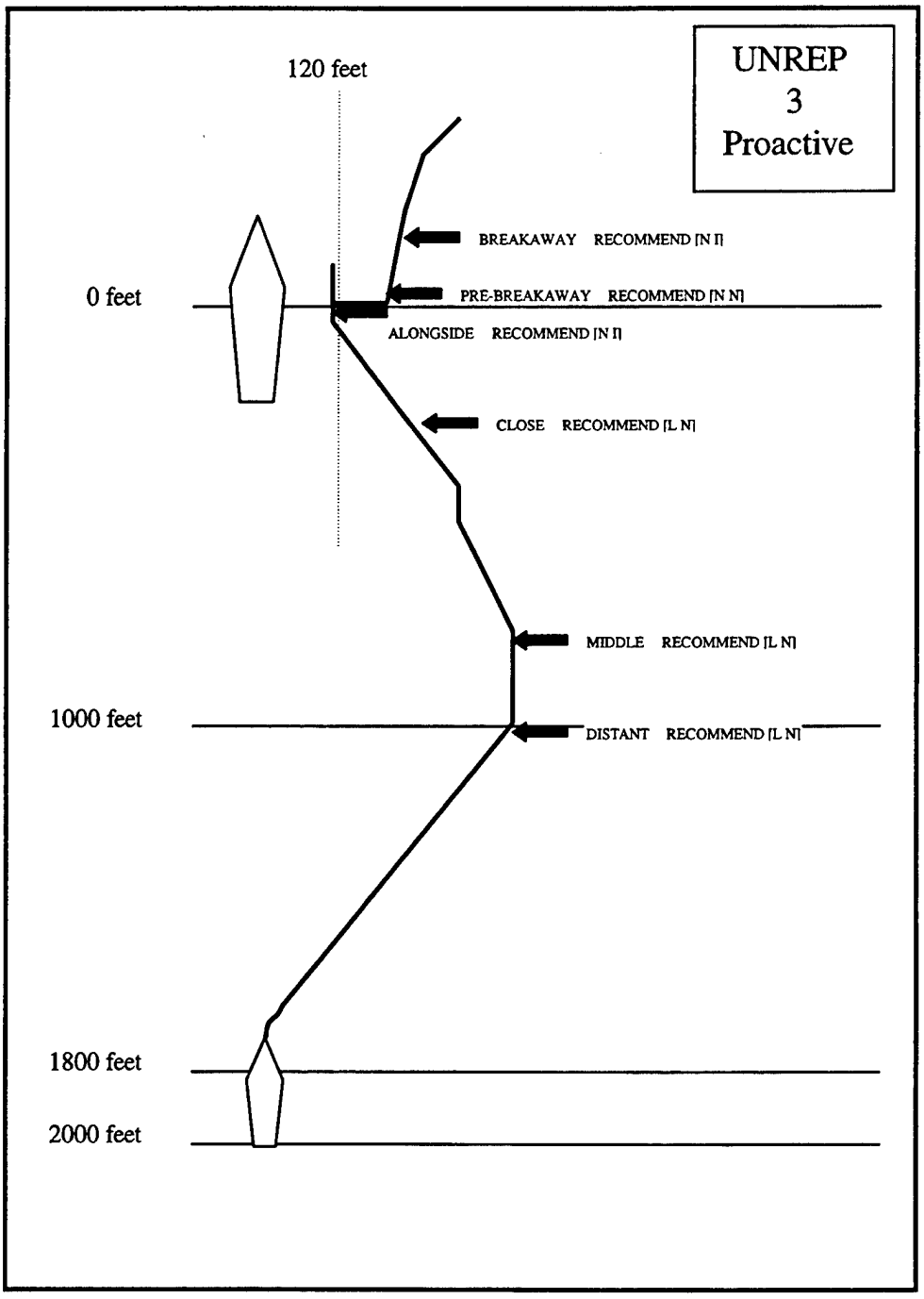

UNREP
4
Aggressive
120 feet
RECOMMEND [N I]
ALONGSIDE
BREAKAWAY RECOMMEND [N N]
PRE-BREAKAWAY RECOMMEND [N D]
0 feet
CLOSE RECOMMEND [N N]
MIDDLE RECOMMEND [R I]
DISTANT RECOMMEND [R N]
1000 feet
1800 feet
2000 feet

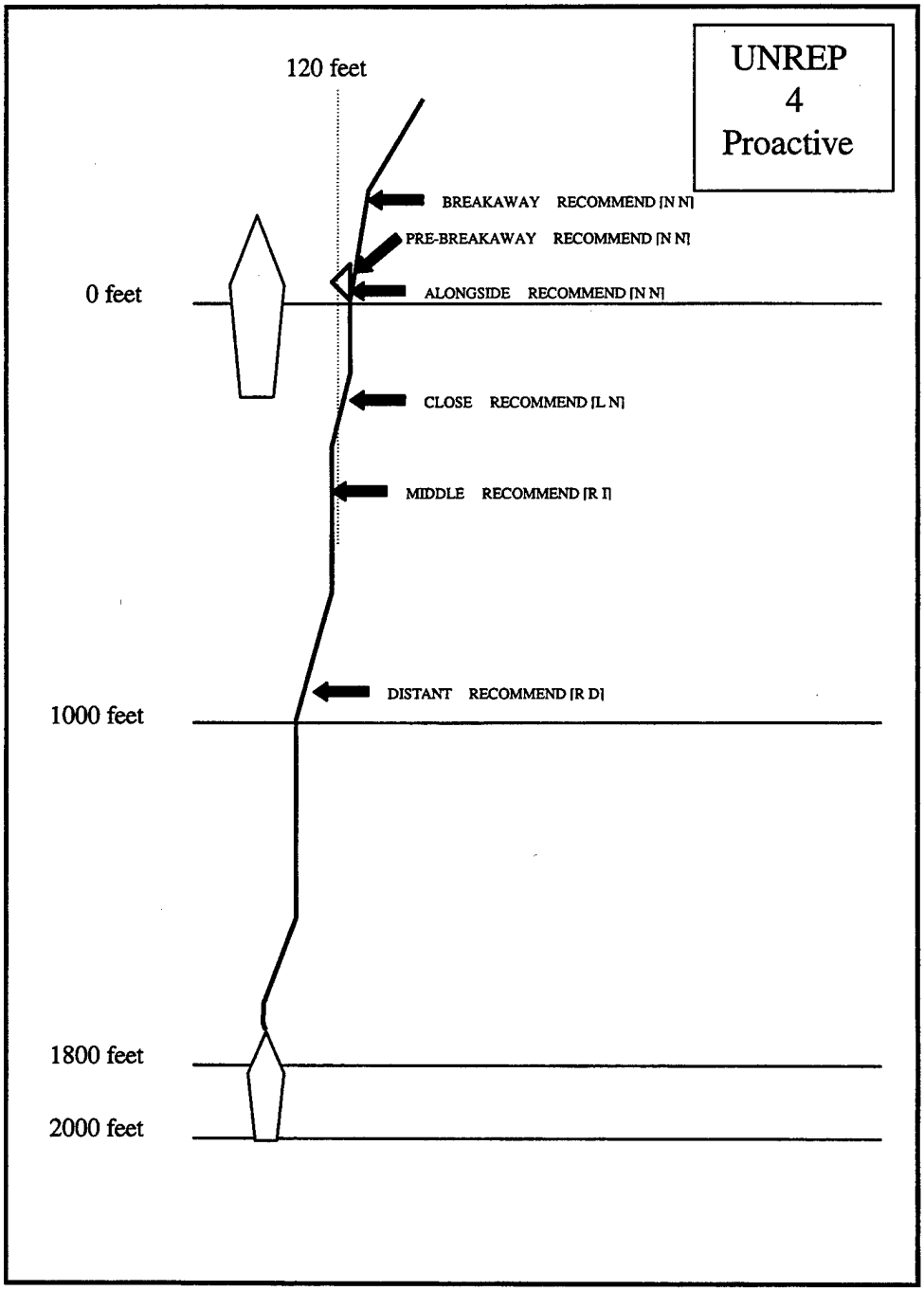
The aggressive CO expected an approach that was as direct as possible. The first run shows at middle approach the CO wanted to come left, even when the lateral separation was just past 120 feet. The second run demonstrates that even though the aggressive CO wants a direct approach, lateral separation must not be less than 120 feet. This CO also wants a fast breakaway, but didn't care if the conning officer did it one or two degrees off replenishment course. The aggressive CO profile is more apt to make a recommended change.

C. THE PROACTIVE TRAINING PROFILE



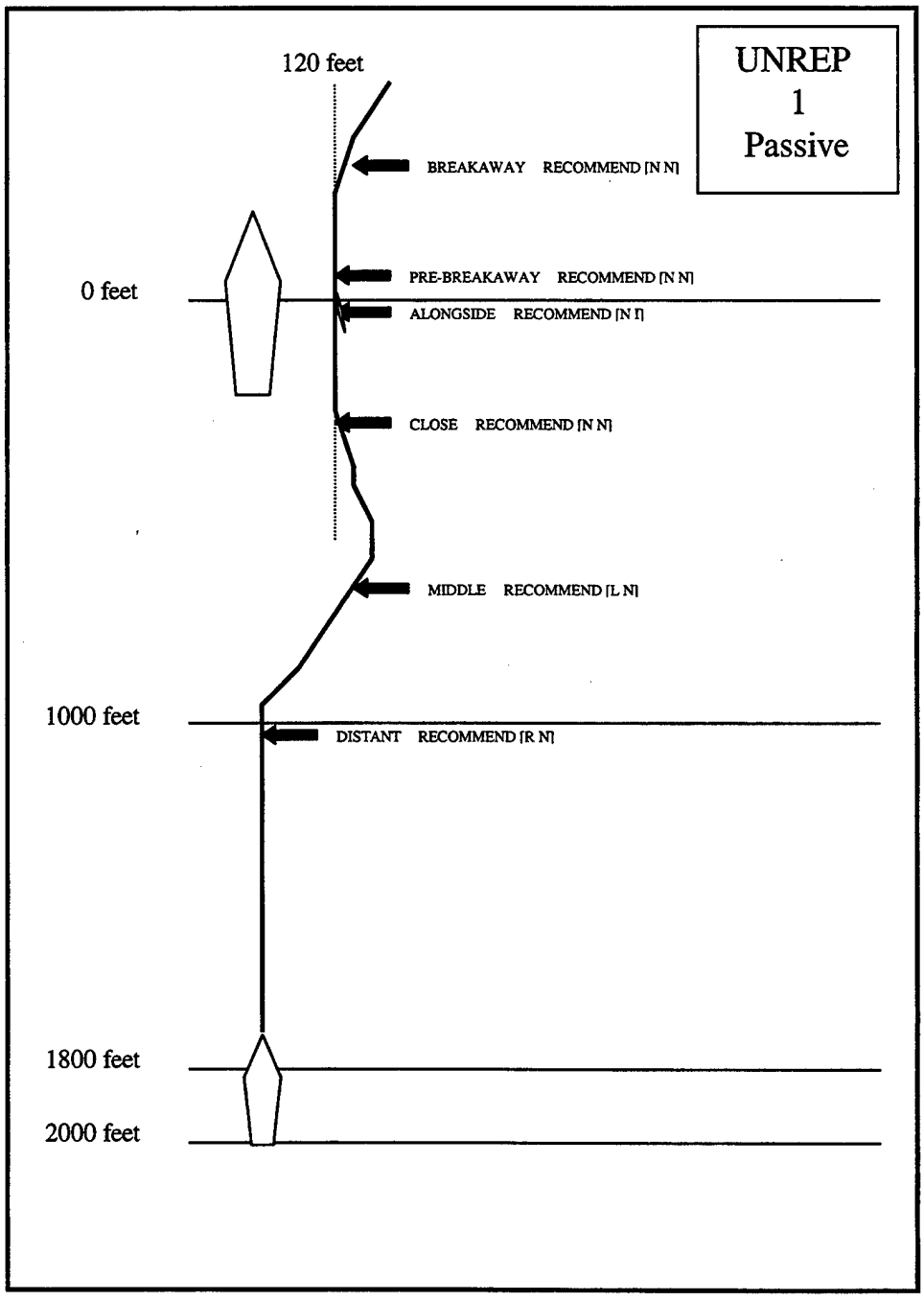

UNREP
2
Proactive
120 feet
RECOMMEND [R D]
ALONGSIDE
BREAKAWAY RECOMMEND [N I]
PRE-BREAKAWAY RECOMMEND [N N]
0 feet
CLOSE RECOMMEND [R N]
MIDDLE RECOMMEND [L N]
1000 feet
DISTANT RECOMMEND [L N]
1800 feet
2000 feet

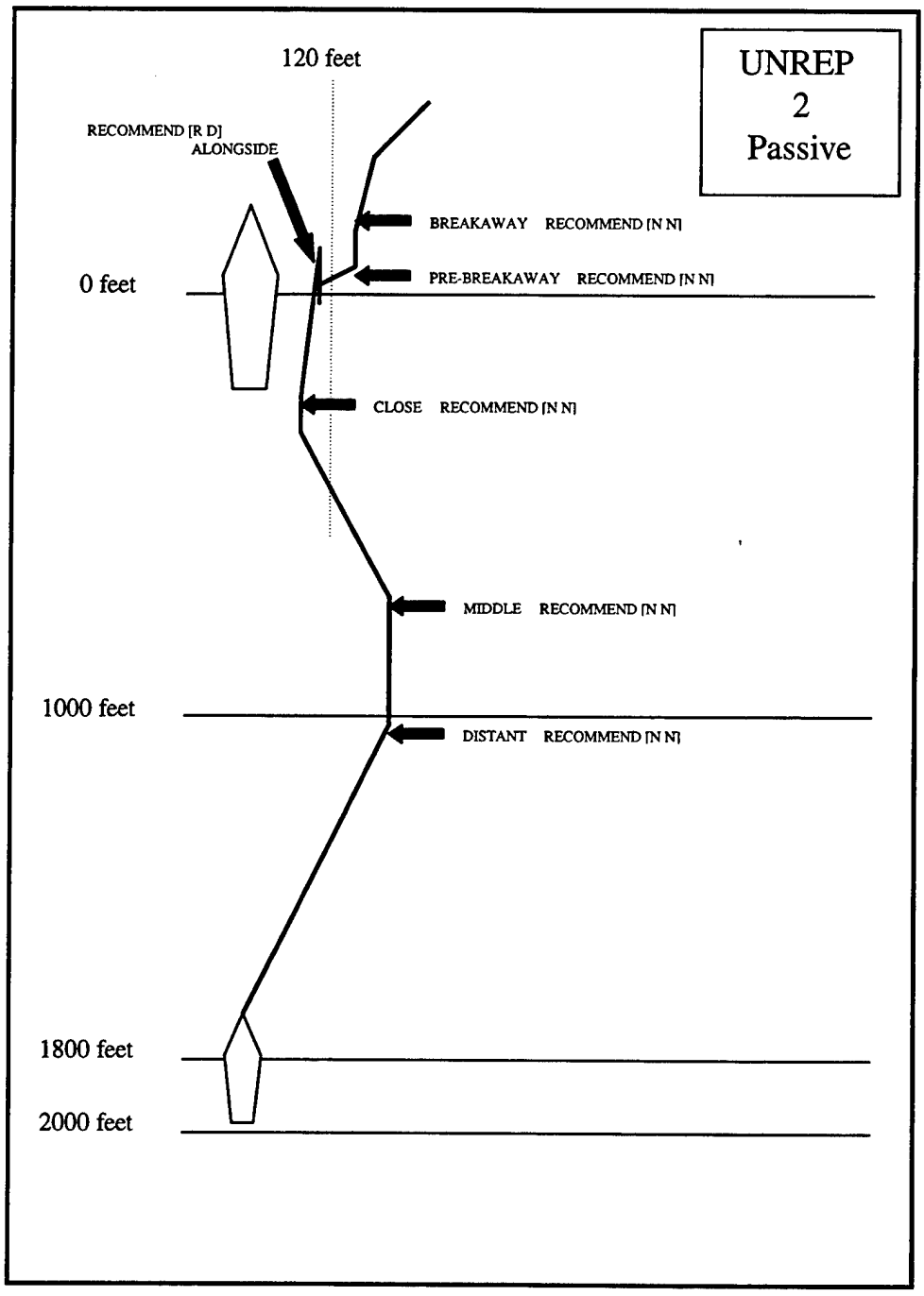

UNREP
3
Proactive
120 feet
0 feet
BREAKAWAY RECOMMEND [N I]
PRE-BREAKAWAY RECOMMEND [N N]
ALONGSIDE RECOMMEND [N I]
CLOSE RECOMMEND [L N]
MIDDLE RECOMMEND [L N]
1000 feet
DISTANT RECOMMEND [L N]
1800 feet
2000 feet

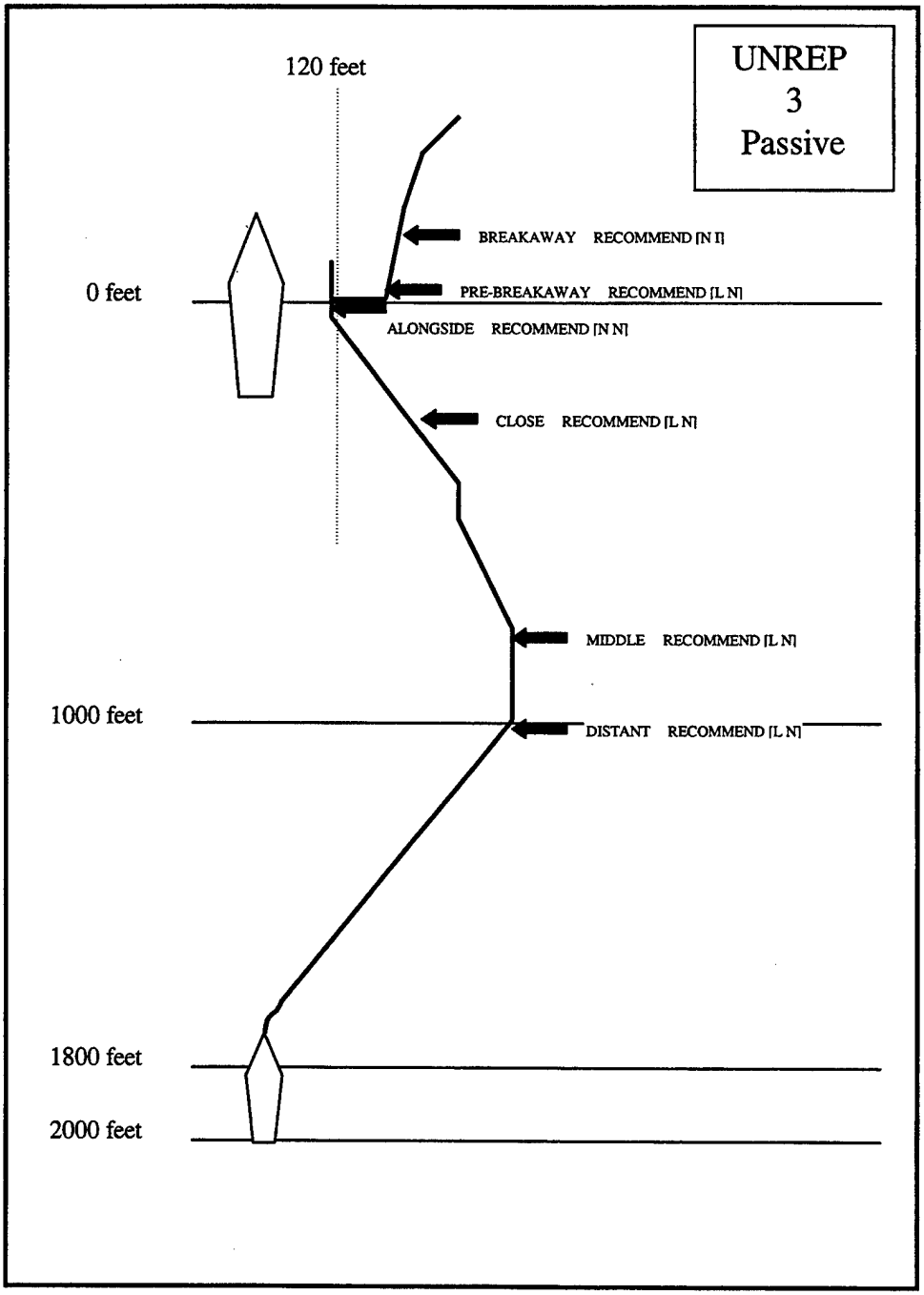

UNREP
4
Proactive
120 feet
BREAKAWAY RECOMMEND [N N]
PRE-BREAKAWAY RECOMMEND [N N]
0 feet
ALONGSIDE RECOMMEND [N N]
CLOSE RECOMMEND [L N]
MIDDLE RECOMMEND [R I]
DISTANT RECOMMEND [R D]
1000 feet
1800 feet
2000 feet

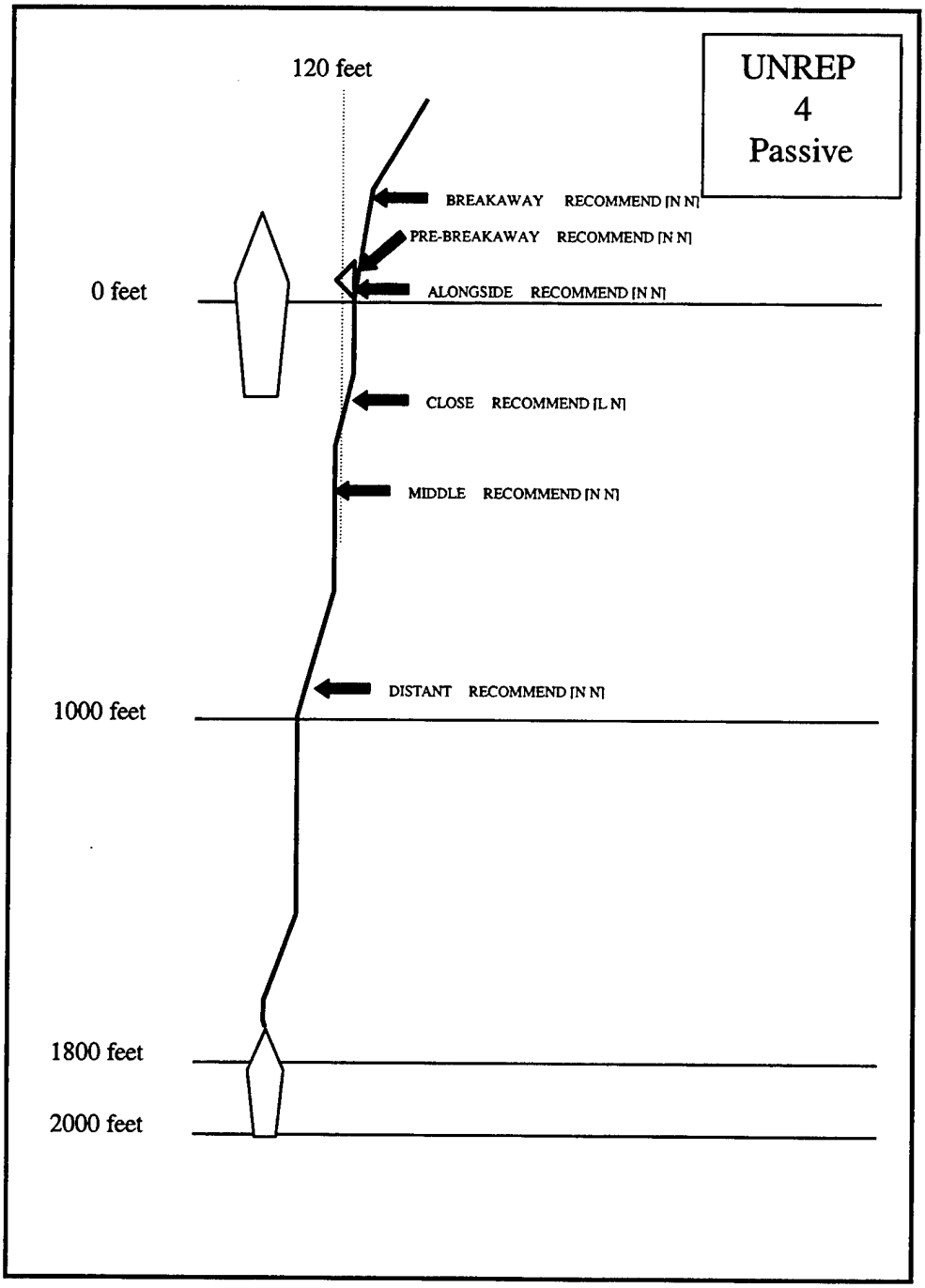
The Proactive CO also expected a fast direct approach, but was less apt to make a change recommendation. This CO also wanted to maintain a safe distance outside 120 feet lateral separation. The proactive CO prefers the conning officer to open the lateral separation before breaking away, to ensure the stern is clear of the control ship's bow.

D. THE PASSIVE TRAINING PROFILE




UNREP
2
Passive
120 feet
RECOMMEND [R D]
ALONGSIDE
BREAKAWAY RECOMMEND [N N]
PRE-BREAKAWAY RECOMMEND [N N]
0 feet
CLOSE RECOMMEND [N N]
MIDDLE RECOMMEND [N N]
1000 feet
DISTANT RECOMMEND [N N]
1800 feet
2000 feet


UNREP
3
Passive
120 feet
0 feet
BREAKAWAY RECOMMEND [N I]
PRE-BREAKAWAY RECOMMEND [L N]
ALONGSIDE RECOMMEND [N N]
CLOSE RECOMMEND [L N]
MIDDLE RECOMMEND [L N]
1000 feet
DISTANT RECOMMEND [L N]
1800 feet
2000 feet


UNREP
4
Passive
120 feet
BREAKAWAY RECOMMEND [N N]
PRE-BREAKAWAY RECOMMEND [N N]
0 feet
ALONGSIDE RECOMMEND [N N]
CLOSE RECOMMEND [L N]
MIDDLE RECOMMEND [N N]
DISTANT RECOMMEND [N N]
1000 feet
1800 feet
2000 feet

The passive CO does prefer a direct approach. This is demonstrated with the recommendation to come left at middle point of runs one and three. However, this CO was not as consistent as the other CO's. Run two shows the conning officer has ended up right of 120 feet lateral separation at the middle point, but the Passive CO didn't make a recommendation to change. This CO was less apt to recommend a change if the lateral separation on approach was less than 120 feet. The passive CO typically did not make course and speed change recommendations during the breakaway. Overall, this profile was less apt to make a change recommendation.

E. PROFILE VALIDATION

With the profiles completed, the remaining goal of the thesis was to have profiles validated by experienced ship-handlers. This goal was accomplished by having two SWO's with higher experience levels individually review the profiles for accuracy. Each participant signed an agreement to participate in the validation anonymously. Participants were given the same initial data given to the video segment review subjects, and asked to review and comment on whether they agreed or disagreed with the recommendations made by the profiled CO for each UNREP run. The review process took approximately one hour per session.

The goals of the validation were as follows:

- Identify any discrepancies in the profiles based on a senior shiphandler's perceptions of an aggressive, proactive, or passive CO's training methodologies.

- Identify any missing crucial or redundant decision points that a CO would have.
- Evaluate the efficiency of this approach to knowledge elicitation for this scenario.

The following is a brief profile of the naval personnel who reviewed the VCO profiles. Both were qualified Surface Warfare Officers. One was a post Executive Officer, and the other was a post Department Head. There was one Commander and one Lieutenant Commander. The Commander is currently a Curriculum Officer at the Naval Postgraduate School, and the Lieutenant Commander is a non-computer science graduate student.

The validation reviews proved to be very useful and resulted in selected revisions to the VCO profiles. The most significant issues raised were that the use of more selected points would result in more accurate profiles, and determining the accuracy of a profile during the alongside phase is difficult to determine with only 20 seconds observed at that point in the environment. Based on this validation, the resulting VCO profiles are examples, with 80 percent accuracy, of three different CO training methodologies used during six points in an UNREP evolution. This level of accuracy was taken from the recommendations that the validation participants considered correct.

V. DEVELOPING A VIRTUAL COMMANDING OFFICER

A. SELECTING A COGNITIVE ARCHITECTURE

The criteria for selecting a cognitive architecture came down to finding a system that is flexible enough to meet the demands of the task and is leading the way in cognitive science research. For many years *Lisp* has been the standard 'list processing' programming language of artificial intelligence and cognitive psychology. However, Lisp was preceeded by other list processors. One such processor, known as the Information Processing Language (IPL), is considered by some, to be the first high level programming language. Allen Newell, known for his work in the field of Artificial Intelligence (AI), was responsible for this language and many other contributions to the cognitive science community. One of his recent achievements was his contribution toward the development of Soar, to many, considered one of the most promising cognitive architectures AI has produced. [MICH92]

Since the Soar project inception in 1983, research interest has grown dramatically. The community consists of over 90 computer science and psychology researchers in the United States and Europe. [ROSE93] Soar is an architecture that is capable of solving a variety of problems by formulating an (internal) goal and subsequently searching its memory for suitable data structures that possess the characteristics determining a problem space. If one is found, Soar attempts to find a data structure within the problem space that matches the current problem state. Finally, Soar will attempt to find operators that allow it to modify the present state in a way that the computed distance from the desired

state (goal) is reduced according to some criterion. If a match cannot be found, Soar enters into an impasse that generates a new sub-goal, and this process continues until a solution to the problem is found. Based on this, the system generates new rules that are added to the existing rules to extend its knowledge. This makes Soar an extremely efficient learner. [MICH92]

Since Soar appears to have a promising future in ITS and is compatible for integration into Lisp, it seems like an appropriate choice for this simple VCO design example. A Lisp example of a means ends analysis and its application to an UNREP VCO ITS is discussed later in this chapter.

B. SOAR-LIKE SPECIFICATION FOR A VCO ITS (UNREP SCENARIO)

Creating profiles of the training methods used by a Commanding Officer while training junior officers during an UNREP was the primary goal of this thesis. These profiles could potentially be applied to an Intelligent Tutor for the COVE project simulator. The following VCO ITS specification was constructed in the spirit of a Soar Cognitive Architecture.

As discussed in chapter three, the junior shiphandler was selected for modeling the student. This is indicated by the title "JUNIOR" associated with the Shiphandler in this specification. The UNREP evolution is broken into three primary phases: approach, alongside, and breakaway. These are represented by the titles: UNREPApproach, UNREPAlongside, and UNREPBreakaway associated with Phase.

After reviewing the UNREP shiphandling evolution, four basic corrective measures were identified: altering position to the left, altering position to the right, increasing speed, and decreasing speed. These four measures are applicable to all shiphandling evolutions. In order to determine whether a correction is required, a reference track must be available for comparison. This reference track is usually based on an accepted standard defined by skilled shiphandlers and often includes style characteristics or preferences decided by the ship's CO. The position of the reference track is represented by an argument followed by the word "Suggested." For instance, leftSuggested means the ship is positioned to the right of the reference track. A recommendation is delivered to the shiphandler through Recommend taking into consideration the Shiphandler student model and the associated training method.

P – Plan: Recommend to the respective shiphandler a problem solution based on current problem state.

D – Difference: Identify the difference of desired state and the respective shiphandler's current state.

A – Action: Identify where the respective officer will be when at the desired state.

1. Aggressive Training Method

RegainTrackLeft (right, leftSuggested, JUNIOR, AGGRESSIVE)

"Recommend correct from position right to position left in phase UNREPApproach"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR) Trainer(AGGRESSIVE),

At(JUNIOR, right), In(right, UNREPApproach),

In(leftSuggested, UNREPApproach),

Phase(UNREPApproach), Method(AGGRESSIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, AGGRESSIVE)

"Recommend correct from position left to position right in phase UNREPApproach"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, left), In(left, UNREPApproach),

In(rightSuggested, UNREPApproach),

Phase(UNREPApproach), Method(AGGRESSIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, AGGRESSIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPApproach"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, current), In(current, UNREPApproach),

In(increaseSuggested, UNREPApproach),

Phase(UNREPApproach), Method(AGGRESSIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, AGGRESSIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPApproach"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, current), In(current, UNREPApproach),

In(decreaseSuggested, UNREPApproach),

Phase(UNREPApproach), Method(AGGRESSIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

RegainTrackLeft (right, leftSuggested, JUNIOR, AGGRESSIVE)

"Recommend correct from position right to position left in phase UNREPAlongside"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR) Trainer(AGGRESSIVE),

At(JUNIOR, right), In(right, UNREPAlongside),

In(leftSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(AGGRESSIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, AGGRESSIVE)

"Recommend correct from position left to position right in phase UNREPAlongside"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, left), In(left, UNREPAlongside),

In(rightSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(AGGRESSIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, AGGRESSIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPAlongside"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, current), In(current, UNREPAlongside),

In(increaseSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(AGGRESSIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, AGGRESSIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPAlongside"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, current), In(current, UNREPAlongside),

In(decreaseSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(AGGRESSIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

RegainTrackLeft (right, leftSuggested, JUNIOR, AGGRESSIVE)

"Recommend correct from position right to position left in phase UNREPBreakaway"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR) Trainer(AGGRESSIVE),

At(JUNIOR, right), In(right, UNREPBreakaway),

In(leftSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(AGGRESSIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, AGGRESSIVE)

"Recommend correct from position left to position right in phase UNREPBreakaway"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, left), In(left, UNREPBreakaway),

In(rightSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(AGGRESSIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, AGGRESSIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPBreakaway"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, current), In(current, UNREPBreakaway),

In(increaseSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(AGGRESSIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, AGGRESSIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPBreakaway"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(AGGRESSIVE),

At(JUNIOR, current), In(current, UNREPBreakaway),

In(decreaseSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(AGGRESSIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

2. Proactive Training Method

RegainTrackLeft (right, leftSuggested, JUNIOR, PROACTIVE)

"Recommend correct from position right to position left in phase UNREPApproach"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, right), In(right, UNREPApproach),

In(leftSuggested, UNREPApproach),

Phase(UNREPApproach), Method(PROACTIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, PROACTIVE)

"Recommend correct from position left to position right in phase UNREPApproach"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, left), In(left, UNREPApproach),

In(rightSuggested, UNREPApproach),

Phase(UNREPApproach), Method(PROACTIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, PROACTIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPApproach"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, current), In(current, UNREPApproach),

In(increaseSuggested, UNREPApproach),

Phase(UNREPApproach), Method(PROACTIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, PROACTIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPApproach"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, current), In(current, UNREPApproach),

In(decreaseSuggested, UNREPApproach),

Phase(UNREPApproach), Method(PROACTIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

RegainTrackLeft (right, leftSuggested, JUNIOR, PROACTIVE)

"Recommend correct from position right to position left in phase UNREPAlongside"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, right), In(right, UNREPAlongside),

In(leftSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(PROACTIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, PROACTIVE)

"Recommend correct from position left to position right in phase UNREPAlongside"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, left), In(left, UNREPAlongside),

In(rightSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(PROACTIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, PROACTIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPAlongside"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, current), In(current, UNREPAlongside),

In(increaseSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(PROACTIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, PROACTIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPAlongside"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, current), In(current, UNREPAlongside),

In(decreaseSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(PROACTIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

RegainTrackLeft (right, leftSuggested, JUNIOR, PROACTIVE)

"Recommend correct from position right to position left in phase UNREPBreakaway"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, right), In(right, UNREPBreakaway),

In(leftSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(PROACTIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, PROACTIVE)

"Recommend correct from position left to position right in phase UNREPBreakaway"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, left), In(left, UNREPBreakaway),

In(rightSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(PROACTIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, PROACTIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPBreakaway"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, current), In(current, UNREPBreakaway),

In(increaseSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(PROACTIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, PROACTIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPBreakaway"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(PROACTIVE),

At(JUNIOR, current), In(current, UNREPBreakaway),

In(decreaseSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(PROACTIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

3. Passive Training Method

RegainTrackLeft (right, leftSuggested, JUNIOR, PASSIVE)

"Recommend correct from position right to position left in phase UNREPApproach"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, right), In(right, UNREPApproach),

In(leftSuggested, UNREPApproach),

Phase(UNREPApproach), Method(PASSIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, PASSIVE)

"Recommend correct from position left to position right in phase UNREPApproach"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, left), In(left, UNREPApproach),

In(rightSuggested, UNREPApproach),

Phase(UNREPApproach), Method(PASSIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, PASSIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPApproach"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, current), In(current, UNREPApproach),

In(increaseSuggested, UNREPApproach),

Phase(UNREPApproach), Method(PASSIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, PASSIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPApproach"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, current), In(current, UNREPApproach),

In(decreaseSuggested, UNREPApproach),

Phase(UNREPApproach), Method(PASSIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

RegainTrackLeft (right, leftSuggested, JUNIOR, PASSIVE)

"Recommend correct from position right to position left in phase UNREPAlongside"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, right), In(right, UNREPAlongside),

In(leftSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(PASSIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, PASSIVE)

"Recommend correct from position left to position right in phase UNREPAlongside"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, left), In(left, UNREPAlongside),

In(rightSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(PASSIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, PASSIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPAlongside"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, current), In(current, UNREPAlongside),

In(increaseSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(PASSIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, PASSIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPAlongside"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, current), In(current, UNREPAlongside),

In(decreaseSuggested, UNREPAlongside),

Phase(UNREPAlongside), Method(PASSIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

RegainTrackLeft (right, leftSuggested, JUNIOR, PASSIVE)

"Recommend correct from position right to position left in phase UNREPBreakaway"

P: Position(right), Position(leftSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, right), In(right, UNREPBreakaway),

In(leftSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(PASSIVE),

Recommend (JUNIOR, leftSuggested)

D: At(JUNIOR, right)

A: At(JUNIOR, leftSuggested)

RegainTrackRight (left, rightSuggested, JUNIOR, PASSIVE)

"Recommend correct from position left to position right in phase UNREPBreakaway"

P: Position(left), Position(rightSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, left), In(left, UNREPBreakaway),

In(rightSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(PASSIVE),

Recommend (JUNIOR, rightSuggested)

D: At(JUNIOR, left)

A: At(JUNIOR, rightSuggested)

RecommendIncreaseSpeed (current, increaseSuggested, JUNIOR, PASSIVE)

"Recommend increase speed from current speed to suggested speed in phase UNREPBreakaway"

P: Speed(current), Speed(increaseSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, current), In(current, UNREPBreakaway),

In(increaseSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(PASSIVE),

Recommend (JUNIOR, increaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, increaseSuggested)

RecommendDecreaseSpeed (current, decreaseSuggested, JUNIOR, PASSIVE)

"Recommend decrease speed from current speed to suggested speed in phase UNREPBreakaway"

P: Speed(current), Speed(decreaseSuggested), Shiphandler(JUNIOR), Trainer(PASSIVE),

At(JUNIOR, current), In(current, UNREPBreakaway),

In(decreaseSuggested, UNREPBreakaway),

Phase(UNREPBreakaway), Method(PASSIVE),

Recommend (JUNIOR, decreaseSuggested)

D: At(JUNIOR, current)

A: At(JUNIOR, decreaseSuggested)

C. MEANS ENDS ANALYSIS GENERAL EXAMPLE

While exploring the Soar cognitive architecture, it became apparent that Lisp would be a practical choice for creating an example means ends analysis. Many of the Soar examples uncovered closely resembled Lisp. Though the first implementations of the language surfaced in the 1950's, it is still in the forefront of programming language technology. After FORTRAN, Lisp is the oldest language still in use. Lisp is ideal for modeling a concept into a prototype that can demonstrate the concept. [GRAH96]

A general machine assembly means ends analysis was developed in Allegro ANSI Common Lisp 4.3 to determine if means ends would be suited for an UNREP shiphandling

task. The machine assembly example is included in Appendix B. Means ends analysis is typically used for general problem solving and ordered sequence tasking. Two common and important variables in means ends are the states that an object can be in. For an UNREP, the object would be ownship and the states are current state and goal state. When placed in a Cartesian coordinate system, the goal state is placed at the origin, and the current state is either equal to the goal state or it is not. Nine basic operators are used to determine where the current state is and how to correct the state if it does not equal the goal. For instance if the current state is in the Ahead and Left quadrant, then the steps to correct would be come right and decrease speed. (Figure 6)

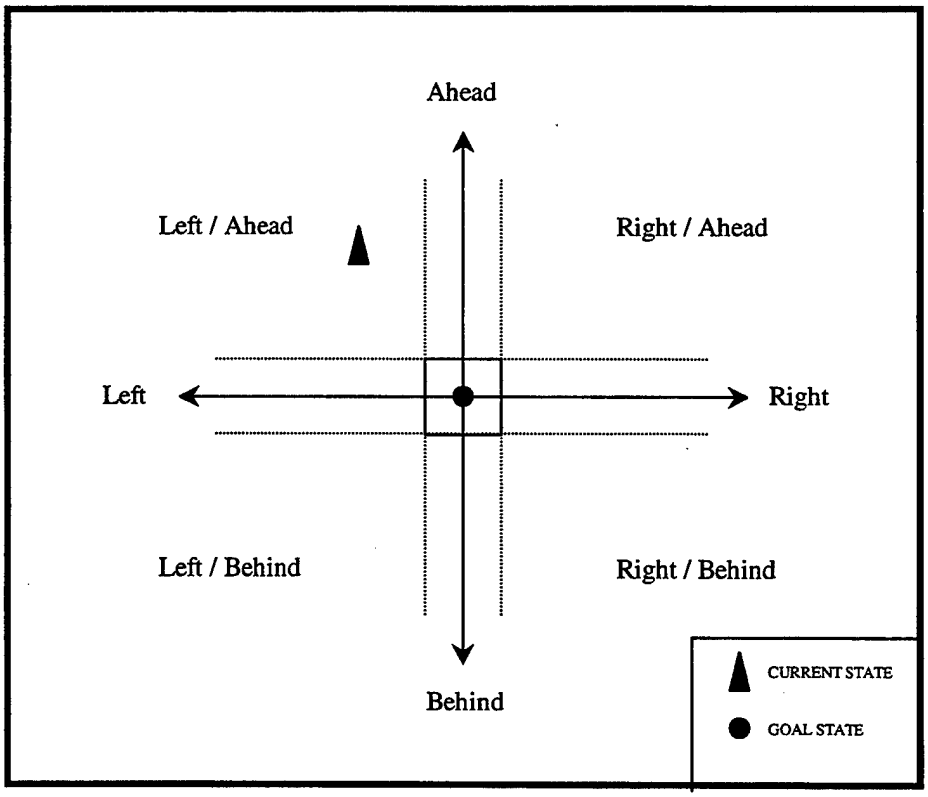


Figure 6: State coordinate system

The general example in Appendix B demonstrates a machine assembly problem that can apply four operators to transverse through states to reach a goal. In order to complete the assembly task and reach the goal state, three components must be assembled in a specific order. A path is selected using a breadth first search algorithm. This example is based on outcomes that are deterministic. For instance, if a component is to be assembled, the assembly is performed completely. Partial completion or work in progress is not a factor. Since many factors apply to the imperfect reality of shiphandling, special considerations would be necessary for a means ends shiphandling application.

The shiphandling example would require five operators: come-left, come-right, increase-speed, decrease-speed, steady-ahead. A ship that is marked left of track and corrects by coming right does not guarantee the ship will automatically regain track. Therefore, the distance the ship is off track is necessary to decide when to stop correcting. One solution is to measure the distance in set increments or units. The number of units used would be directly related to the desired level of accuracy. For example, a ship could be located at one of four units to the right of track, one of four units to the left of track, or at a unit on track. This makes nine possible states just to determine a ships position left or right. When ahead or behind track is incorporated with the same number of units, the number of states expands to eighty-one. Going a step further, since rudder and engine orders affect the motion of the ship over time, the means ends analysis needs to involve time. Using more rudder and speed is likely to correct the ship's position quicker.

Using means ends analysis for an UNREP VCO ITS appears to have potential. The considerations mentioned above might only be a subset of what is required to ensure the accuracy of a shiphandling tutor.

VI. CONCLUSIONS

A. SUMMARY OF WORK

The skill of training a junior officer in the art of shiphandling is a valuable one. The simulators available today require the presence of a human expert to train. This makes the simulator costly, not very portable, and limits the number of ship drivers that can benefit. The Surface Warfare Officers School (SWOS) has recognized what the advances in VE technology can do to enhance the way they train Surface Warriors. Researchers at the Naval Air Warfare Center Training Systems Division (NAWC TSD) have developed a test-bed simulator called the Conning Officer Virtual Environment (COVE). COVE currently models an Underway Replenishment (UNREP) scenario, but it does not provide expert training feedback. The purpose of this thesis was to identify three training methodologies practiced by Commanding Officers during an UNREP and to develop those methodologies into profiles. In addition, this thesis provides some examples of how a Virtual Commanding Officer could be integrated into the UNREP shiphandling VE. As the development of the COVE project advances, the need to maintain the collaboration between experienced ship-handlers and researchers is necessary. The research for this thesis was done and reviewed by experienced ship-handlers.

The design of an Intelligent Tutoring System for the VE shiphandling test-bed is work in progress. This thesis proposes and demonstrates one approach to designing such a system that can be adapted to varying trainer profiles. After a thorough training review

and a shiphandling background survey, three training methodologies were identified as being most commonly practiced by Commanding Officers. Varying combinations of the three methodologies are also practiced. Thus, a simple yet dynamic system is required to meet the demand for flexibility. Establishing unique personal profiles is a logical way to distinguish between the variety. Ultimately, this system would allow a Commanding Officer to create a self-profiled Virtual Commanding Officer, based on their personal expectations.

The first step in profile creation was to determine at what point to make an evaluation of a situation for providing direction. The profiles created in this thesis are based on only six points in an UNREP evolution. The points in between are infinite. Therefore, before the intelligent tutor can be fully implemented, a decision of the frequency necessary to effectively evaluate the status of an evolution must be determined. After careful review, evaluation at six points was determined sufficient to demonstrate the creation of UNREP scenario profiles. The next step was to determine how many basic corrections could be directed. Five corrections were identified for UNREP and were found to be common to most, if not all, shiphandling evolutions.

After domain knowledge in the form of recommendations was collected and compiled, the four example UNREP runs were traced marking the six evaluation points on each run. The compiled recommendations from the three CO profiles were then applied at each point.

Finally, in an effort to do quality assurance and to validate the VCO profiles, the profiles were presented to two experienced Surface Warfare Officers. They were

instructed to review the profiles paying particular attention to personality and style they would expect from that profile. The information gathered during the validation interviews was used to improve the accuracy of the profiles and the approach to collecting the domain expertise. This validation process ensured that subjects with higher levels of experience in shiphandling had reviewed the profiles and that the profiles would be as accurate as possible.

The result of the research was a set of three validated Virtual Commanding Officer profiles for an UNREP. The sample integration with a Soar-like specification and the proposal of a Lisp means ends analysis was produced to demonstrate how the profiles could be implemented.

B. THESIS QUESTIONS

The following questions were addressed in this thesis:

- *What specific training methodologies are most commonly used by CO's during an UNREP evolution?*

  The results of a survey showed that most SWO's agreed, a combination of the three profiles addressed in this thesis is most common. Most CO's expect an aggressive approach, and their training methodology is directly related. While alongside, style is not a factor. Therefore, the CO seems to take on a proactive training profile, planning ahead by anticipating the motion between the ships. The breakaway profile is most likely decided by the mood of the CO once the evolution is complete. This question was addressed in chapter 3.

- *How can an Intelligent Tutoring System be used to improve current training effectiveness?*

  Feedback to the shiphandler is important to correct poor habits and to reinforce good habits. The current training does not provide this feedback. The ITS can ensure the shiphandler walks away with something useful, by producing a post run performance report that can be printed and held for reference. This is one way to ensure the student walks away a better shiphandler, not a better virtual shiphandler. This question was addressed in chapter 2.

- *Which cognitive architecture is best suited for integration into a shiphandling simulator?*

  Determining which is best is difficult to answer. There are a number of architectures available today for developing tutoring systems. Soar was selected because it is a cognitive architecture with a promising future. It is in the spotlight of cognitive research, and is currently being used to develop the Steve animated pedagogical agent. The Air Force Research Laboratory is already funding the continued development of Steve into a usable instructional tool in collaboration with other Intelligent Systems Technology. Integration into a shiphandling simulator would probably be a simple task. This question was addressed in chapter 5.

C. RECOMMENDATIONS FOR FUTURE WORK

1. Future Scenarios

The following is a list of future training scenarios for profiling CO's:

- Maneuvering in restricted waters (harbor transits)
- Maneuvering in shallow water
- Maneuvering through high traffic areas
- Minefield maneuvering
- Plane guard (stationed 1000 yards behind an Aircraft Carrier)
- Division tactics (formation steaming with one or more additional ships)
- Anchoring
- Man overboard maneuvers (Williamson, Anderson, and Y-back turns)
- Approaching a pier (with or without tugs)

2. Commanding Officer Self-profile Interface

A Commanding Officer will most likely not agree with any one profile integrated into a tutoring system. The CO self-profile interface would be a tool for creating the tutor's profile that they agree with. For instance it could use a pre-designed form with radio buttons to select a desired approach. Use of the radian rule lateral separation parameter, define approach speed, and define breakaway heading are all examples of potential selectable criteria to use in this interface.

3. Expertise Analysis

Determining who the expert is and capturing that knowledge in a systematic way still needs to be accomplished. Potential thesis work is available to develop a way to analyze a task and determine what makes an expert an expert.

4. Naval Postgraduate School (NPS) Collaboration

Recent discussions between NPS researchers and researchers from NAWCTSD, identified the value of collaboration. NPS offers a unique environment for ship-handling simulator development and testing. In general, working together with NAWCTSD would provide NPS students with new, military relevant research topics, and would provide NAWCTSD researchers access to the rich domain knowledge that NPS students have to offer. These are largely untapped resources. Steps are being taken to bring NPS and NAWCTSD researchers together in a way that both equally benefit.

D. HOW TO USE THE VCO PROFILES

These profiles can be used for development of the COVE UNREP tutoring system. It provides three different views at how Commanding Officers can train during an UNREP, and how a Virtual Commanding Officer could do the same. The use of these profiles to develop simulator scenarios does not guarantee training accuracy. However, it is relatively accurate at modeling a CO's training methodology. Virtual shiphandlers will benefit from a system that provides them accurate feedback on their performance, and

potentially send them back better shiphandlers. A system that models a CO accurately will make the experience more engaging.

APPENDIX A: EXPERIMENT MATERIALS

A. UNREP TEACHING METHODOLOGY EXPERIMENT WORKSHEET

1. Phase One: Collecting Aggressive Response

2. Phase Two: Collecting Proactive Response

3. Phase Three: Collecting Passive Response

B. SCENARIO DATA

1. Stationing Alignment

2. Last Ordered Standard Commands

3. Engine RPM Table

C. EXPERIMENT DOCUMENTATION

1. Participant ReadMe

2. Participant Consent Form

3. Shiphandling Background Questionnaire

Recommended Course and Speed Changes (1)

APPROACH			ALONGSIDE	PRE-	BREAKAWAY
DISTANT	MIDDLE	CLOSE		BREAKAWAY	
1 COURSE R L SPEED I D	2 COURSE R L SPEED I D	3 COURSE R L SPEED I D	4 COURSE R L SPEED I D	5 COURSE R L SPEED I D	6 COURSE R L SPEED I D
7 COURSE R L SPEED I D	8 COURSE R L SPEED I D	9 COURSE R L SPEED I D	10 COURSE R L SPEED I D	11 COURSE R L SPEED I D	12 COURSE R L SPEED I D
13 COURSE R L SPEED I D	14 COURSE R L SPEED I D	15 COURSE R L SPEED I D	16 COURSE R L SPEED I D	17 COURSE R L SPEED I D	18 COURSE R L SPEED I D
19 COURSE R L SPEED I D	20 COURSE R L SPEED I D	21 COURSE R L SPEED I D	22 COURSE R L SPEED I D	23 COURSE R L SPEED I D	24 COURSE R L SPEED I D

Pre-UNREP Evolution Planning

Briefly describe how you expect an UNREP to be performed.
(Please include Course headings, Ranges, Speeds, Visual Cues, Etc.)

Approach:

Alongside:

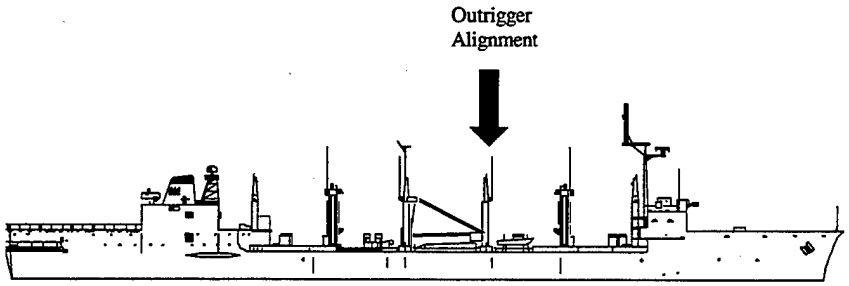
Breakaway:

Recommended Course and Speed Changes (2)

APPROACH			ALONGSIDE	PRE-	BREAKAWAY
DISTANT	MIDDLE	CLOSE		BREAKAWAY	
1 COURSE R L SPEED I D	**2** COURSE R L SPEED I D	**3** COURSE R L SPEED I D	**4** COURSE R L SPEED I D	**5** COURSE R L SPEED I D	**6** COURSE R L SPEED I D
7 COURSE R L SPEED I D	**8** COURSE R L SPEED I D	**9** COURSE R L SPEED I D	**10** COURSE R L SPEED I D	**11** COURSE R L SPEED I D	**12** COURSE R L SPEED I D
13 COURSE R L SPEED I D	**14** COURSE R L SPEED I D	**15** COURSE R L SPEED I D	**16** COURSE R L SPEED I D	**17** COURSE R L SPEED I D	**18** COURSE R L SPEED I D
19 COURSE R L SPEED I D	**20** COURSE R L SPEED I D	**21** COURSE R L SPEED I D	**22** COURSE R L SPEED I D	**23** COURSE R L SPEED I D	**24** COURSE R L SPEED I D

Replenishment Oiler
(Control Ship)

Deliver petroleum and munitions simultaneously to carrier battle groups using both connected and vertical replenishment.



Wichita (AOR 1) Class

Displacement:	41,350 tons
Length:	659 ft.
Beam:	96 ft.
Max Speed:	20 knots
Complement:	460

UNREP Video Segment Data
LAST STANDARD COMMAND ISSUED
Romeo Corpen 130 / Romeo Speed 15

APPROACH			ALONGSIDE	PRE-	BREAKAWAY
DISTANT	MIDDLE	CLOSE		BREAKAWAY	
1 C 130 S 20	2 Right 15° S 20	3 C 128 S 20	4 C 130 S 15	5 C 131 S 070 RPMS	6 C 132 S 25
7 C 135 S 20	8 C 130 S 20	9 C 130 S 20	10 C 130 S 067 RPMS	11 C 131 S 074 RPMS	12 C 132 S 072 RPMS
13 C 140 S 20	14 C 130 S 20	15 C 128 S 20	16 C 131 S 062 RPMS	17 C 132 S 074 RPMS	18 C 135 S 074 RPMS
19 C 131 S 25	20 C 130 S 087 RPMS	21 C 130 S 087 RPMS	22 C 130 S 074 RPMS	23 C 131 S 072 RPMS	24 C 131 S 25

ENGINE RPM TABLE

The model ship in this scenario is a CG-47 class Cruiser, it has four gas turbine engines, two screws and two rudders. They cannot be used independently in this simulation. The following bell schedule will used for the model. It is important to note that in this simulation, engine bells are limited to increments of five knots when pitch is less than 100%. Fine adjustments can be made by changing the RPMs.

AHEAD				AHEAD	
KNOTS	RPMs	PITCH (%)	KNOTS	RPMs	PITCH (%)
1/3				**FLANK 1**	
1	055	7	21	101	100
2	055	13	22	106	100
3	055	20	23	111	100
4	055	27	24	116	100
5	055	34	25	121	100
2/3				**FLANK 2**	
6	055	42	26	126	100
7	055	49	27	131	100
8	055	57	28	138	100
9	055	67	29	146	100
10	055	80	30	153	100
STD				**ASTERN**	
11	055	90	1/3	055	-49
12	055	100	2/3	085	-49
13	061	100	FULL	120	-49
14	066	100			
15	072	100			
FULL					
16	076	100			
17	081	100			
18	086	100			
19	091	100			
20	097	100			

Subject Readme
Virtual Commanding Officer Experiment

Before getting started, I wish to thank you for volunteering to participate in this study. You are about to participate in an experiment designed to evaluate training methodologies and interactions between a Commanding Officer and a junior shiphandler. You have been selected from a group of qualified Surface Warfare Officers. Your comments during a video segment review will be recorded for the purpose of creating CO profiles for a VCO Intelligent Tutoring System. This experiment is not meant to grade your proficiency in shiphandling, and your individual comments will be held strictly confidential.

This experiment should take approximately 50 minutes.

Before we begin, do you have any questions?

Scenario:
For purposes of this experiment, you are tasked to train a junior officer during an Underway Replenishment. The scenario has the junior shiphandler making an approach on the starboard side of a Wichita Class oilier.

Consent Form
Virtual Commanding Officer Experiment

You are invited to participate in a study of the Virtual Commanding Officer (VCO) Intelligent Tutoring System (ITS). With data from you and other participants I hope to illicit knowledge about training methodology associated with a shiphandling task, specifically Underway Replenishment (UNREP). I ask that you read and sign this form indicating that you agree to be in the study. Please ask any questions you may have before signing.

Background information: This study is being conducted for the purpose of evaluating shiphandling training methodologies practiced by Commanding Officers.

LT Karl Tenney, USN (*tenneykr@cs.nps.navy.mil*)

Risks and benefits of being in the study: This study has no unordinary risks beyond those encountered in your everyday workplace. The benefit to participants is that they gain exposure to the Conning Officer Virtual Environment test-bed simulator, a VE simulator with a promising future for training shiphandling skill.

Compensation: No tangible rewards will be given. If requested, a copy of the results can be made available to you at the conclusion of the experiment.

Confidentiality: The records of this study will be kept private. No information will be made publicly accessible that might make it possible to identify you as a participant.

Voluntary nature of the study: If you decide to participate, you are free to withdraw at any time without prejudice.

Statement of consent: I have read the above information. I have asked questions and have had my questions answered. I consent to participate in the study.
If requested, a copy of this form can be provided for your records.

Signature	Date	e-mail

Signature of Administrator	Date

Shiphandling Background
Questionnaire

1. Which category best represents the number of years you have served in the Navy?

   Less than 1 2-3 4-9 10-14 15 or more

2. Of that time which category best represents your time on sea duty?

   None Under 1 year 1-3 years 4-9 years 10 or more

3. Before becoming a commissioned officer, did you receive any shiphandling experience?

   YES NO

   If you circled yes, please give the type of ship and an approximate length of time onboard:

4. On which class of ship have you spent most of your time underway?

5. How many different Commanding Officers have you worked for while on sea duty?

6. For the majority, which of the following best characterizes the shiphandling training methods performed by your CO's:

   A. Observe and Evaluate/let you drive
   B. Quick to Correct/constructive feedback
   C. Discuss Plans and Expectations Before Evolution/critical moment feedback
   D. A Combination of the Above
   E. Other: _____

7. From question 6, which method had the best training effect on you?

8. Which of the following had the most influence on your shiphandling experience: (You may circle 2 choices.)

   A. Observing others perform during an evolution
   B. Performing the evolution
   C. Performing the evolution in a simulated environment
   D. All of the Above
   E. Other: _____

9. During your career, approximately how many UNREP approaches have you performed?

   None 1-3 4-8 9-10 More than 10

10. During your career, approximately how many Harbor Transits have you performed?

    None 1-3 4-8 9-10 More than 10

11. During your career, approximately how many Anchorage/Mooring evolutions have you performed?

    None 1-3 4-8 9-10 More than 10

12. If presented with a self-operated shiphandling simulator, which of the following categories would best represent your purpose for use:

    A. Learning Basic Skills
    B. Refreshing Experience
    C. Improving Skill
    D. A Combination of the Above

13. Has this questionnaire prompted any other comments? Please feel free to use the space below to include them... Your comments and participation in this questionnaire are greatly appreciated!

APPENDIX B: MEANS ENDS ANALYSIS IN LISP

A. MEANS ENDS ANALYSIS IN LISP

Professor Robert McGhee from the Computer Science Department at the Naval Postgraduate School wrote the following Lisp code:

```
(defun make-operator (operator-name preconditions additions deletions)
  (list operator-name preconditions additions deletions))

(defun preconditions (operator) (second operator))

(defun additions (operator) (third operator))

(defun deletions (operator) (fourth operator))

(defun apply-operators (operators state)
  (if (null operators) nil
     (let ((operator (first operators)))
       (if (applicablep operator state)
          (cons (next-state operator state)
              (apply-operators (rest operators) state))
          (apply-operators (rest operators) state)))))

(defun applicablep (operator state)
 (subsetp (preconditions operator) state :test #'equal))

(defun next-state (operator state)
 (union (additions operator)
       (set-difference state (deletions operator) :test #'equal)
       :test #'equal))

(defun successor-list (state-list operator-list)
```

```
  (if (null state-list) nil
     (union (apply-operators operator-list (first state-list))
          (successor-list (rest state-list) operator-list))))

(defun prune (successor-list visited-states-list)
 (if successor-list
  (let ((pruned-list nil) (visited-states visited-states-list))
     (dolist (successor successor-list pruned-list)
          (when (not (already-visitedp successor visited-states))
               (push successor pruned-list)
               (push successor visited-states))))))

(defun already-visitedp (successor-state visited-states-list)
 (member successor-state visited-states-list :test #'equivalent-setp))

(defun equivalent-setp (list1 list2)
 (if (subsetp list1 list2 :test #'equal)
    (if (subsetp list2 list1 :test #'equal) t)))

(defun remove-equivalent-sets (list list-of-lists)
 (if list-of-lists
    (if (equivalent-setp list (first list-of-lists))
       (remove-equivalent-sets list (rest list-of-lists))
       (cons (first list-of-lists)
            (remove-equivalent-sets list (rest list-of-lists))))))

(defun remove-duplicate-sets (list-of-lists)
 (if list-of-lists
    (cons (first list-of-lists)
        (remove-duplicate-sets
         (remove-equivalent-sets (first list-of-lists) list-of-lists)))))

(defun goalp (goal-state state-list)
 (if state-list
```

```
    (if (equivalent-setp goal-state (first state-list)) t
      (goalp goal-state (rest state-list)))))

(defun edited-successor-list (state-list operator-list)
 (remove-duplicate-sets (successor-list state-list operator-list)))

;The following function accomplishes the same thing as the "bfs" function on
;pg. 305 of Dean et al., except it returns the search depth rather than just t,
;and also prunes the search tree to eliminate expansion of previously visited
;states.

(defun goal-search-depth (start-state goal-state operator-list)
 (do* ((depth 1 (1+ depth)) (visited-states-list nil)
     (start start-state) (operators operator-list) (goal goal-state)
     (successor-list (edited-successor-list (list start) operators)
                (prune (edited-successor-list successor-list operators)
                    visited-states-list)))
    ((or (null successor-list) (goalp goal successor-list))
     (if (goalp goal successor-list) depth))))

(defun goal-depthp (start-state goal-state operator-list depth)
 (= depth (goal-search-depth start-state goal-state operator-list)))

(defun applicable-first-operator-list (start-state operator-list)
 (let ((applicable-operator-list nil))
    (dolist (operator operator-list applicable-operator-list)
        (if (applicablep operator start-state)
          (push operator applicable-operator-list)))))

(defun optimal-first-operator-list (start-state goal-state operator-list)
 (let
  ((optimal-list nil) (operator-successor-list nil)
   (operators (applicable-first-operator-list start-state operator-list))
   (depth (1- (goal-search-depth start-state goal-state operator-list))))
```

```
   (dolist (operator operators optimal-list)
        (setf operator-successor-list
            (apply-operators (list operator) start-state))
        (if (goal-attainablep operator-successor-list goal-state
                                        operator-list depth)
          (push operator optimal-list)))))

(defun goal-attainablep (start-state-list goal-state operator-list depth)
 (if start-state-list
    (if (goal-depthp (first start-state-list) goal-state operator-list depth)
       t
       (goal-attainablep (rest start-state-list)
                goal-state operator-list depth))))

(defvar *start-state*
    '((not (assembled 1)) (not (assembled 2)) (not (assembled 3))))

(defvar *goal-state*  '((assembled 1)(assembled 2)(assembled 3)))

(defvar *operator-list*
 (list (make-operator 'a '((not (assembled 1)))
                '((assembled 1))
                '((not (assembled 1))))
     (make-operator 'b '((not (assembled 1)))
                '((assembled 2))
                '((not (assembled 2))))
     (make-operator 'c '((assembled 1))
                '((assembled 2) (not (assembled 1)))
                '((not (assembled 2)) (assembled 1)))
     (make-operator 'd '((assembled 1) (assembled 2))
                '((assembled 3))
                '((not (assembled 3))))))

(defun test1 () (apply-operators *operator-list* *start-state*))
```

```
(defun test2 () (goal-search-depth *start-state* *goal-state* *operator-list*))

(defun test3 () (goal-depthp *start-state* *goal-state* *operator-list* 3))

(defun test4 () (goal-depthp *start-state* *goal-state* *operator-list* 2))

(defun test5 () (applicable-first-operator-list *start-state* *operator-list*))

(defun test6 () (optimal-first-operator-list *start-state* *goal-state*
                                             *operator-list*))
```

LIST OF REFERENCES

[ALES91] Alessi, S. M., Trollip, S. R., *Computer-Based Instruction*, Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1991.

[COXB98] Cox, B., Brunel University, "An Intelligent Interface for Simulation Design," http://www.brunel.ac.uk/~gssrjis/issue/j515.html (HTML Document) 1998.

[GRAH96] Graham, Paul, *ANSI Common Lisp*, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1996.

[KEAR87] Kearsley, Greg, *Artificial Intelligence and Instruction, Applications and Methods*, Reading, Massachusetts, Addison-Wesley Publinshing Company, 1987.

[LAND99] Landauer, Christopher, Bellman, Kirstie L., "Virtually Collaborating with Pedagogical Agents," *Proceedings of the Virtual Worlds and Simulation Conference (VWSIM '99)*, San Diego, California, Simulation Councils, Inc., 1999.

[LOCK98] Lockheed Martin Artificial Intelligence Center, "Intelligent Tutoring Systems," http://vet.parl.com/~vet/studio/itsSystems.html (HTML Document), 1998.

[MAND88] Mandl, H., Lesgold, A., *Learning Issues for Intelligent Tutoring Systems*, New York, New York: Springer-Verlag New York Inc., 1988.

[MICH92] Michon, John A., Lkyurek, Aladin, *Soar: A Cognitive Architecture In Perspective*, Dordrecht, The Netherlands: Kluwer Academic Publishers, 1992.

[POOL98] Poole, Michele, "Building Surface Warriors," *The U.S. Naval Institute Proceedings*, Annapolis, Maryland, U.S. Naval Institute, May 1998.

[RICK97] Rickel, Jeff, "Statement on Pedagogical agents," http://cs.uwp.edu/staff/haller/Activities/SSS97/Statements/rickel.html (HTML Document), 1997.

[ROSE93] Rosenbloom, Paul S., Laird, John E., Newell, Allen, *The Soar Papers Volume 1*, United States: Massachusetts Institute of Technology, 1993.

[SLEE82] Sleeman, D., Brown, J.S., *Intelligent Tutoring Systems*, London: Academic Press Inc., 1982.

[WARR98] Warren, J. R., Li, S., University of South Australia Advanced Computing Research Centre, "An Intelligent Authoring Shell (IAS) for Adaptive Tutoring Systems," http://www.cis.unisa.edu.au/acrc/is/its.html (HTML Document), 1998.

[WENG87] Wenger, Etienne, *Artificial Intelligence and Tutoring Systems*, Los Altos, California: Morgan Kaufmann Publishers, Inc., 1987.

BIBLIOGRAPHY

Chief of Naval Operations, *Underway Replenishment,* Naval Warfare Publication, NWP 4.01.4, August 1996.

Crenshaw, *R. S., Naval Shiphandling*, Annapolis, Maryland: Naval Institute Press, 1975.

Dean, T., Allen, J., Aloimonos, Y., *Artificial Intelligence, Theory and Practice*, Menlo Park, California: Addison-Wesley Publishing Company, 1995.

Frasson, C., Gauthier, G., Lesgold A., *Intelligent Tutoring Systems – Third International Conference, ITS '96 Montreal, Canada, June 1996 Proceedings*, Berlin: Springer Verlag, Berlin Heidelberg, 1996.

Frasson, C., Gauthier, G., McCalla, G. I., *Intelligent Tutoring Systems – Second International Conference, ITS '92 Montreal, Canada, June 1992 Proceedings*, Berlin: Springer-Verlag, Berlin Heidelberg, 1992.

Stavridis, James, *Watch Officers Guide*, Annapolis, Maryland: Naval Institute Press, 1992.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
   8725 John J. Kingman Road, Ste 0944
   Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library.....2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, CA 93943-5101

3. Chairman, Code CS 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Rudolph P. Darken. Code CS/Dr.....1
   Computer Science Department
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Chief of Naval Operations (N62) 1
   2000 Navy Pentagon
   Washington DC 20350-2000

6. Capt. Jay Kistler, USN (N6M) 1
   N6M
   2000 Navy Pentagon
   Room 4C445
   Washington DC 20350-2000

7. George Phillips.....1
   CNO, N6M1
   2000 Navy Pentagon

Room 4C445
Washington DC 20350-2000

8. Terry Allard.....1
Office of Naval Research (342)
800 No. Quincy Street
Arlington, VA 22217-5660

9. Dave Munroe.....1
Surface Warfare Officers School Command
446 Cushing Road
Newport, RI 02841-1209

10. Capt. Dennis McBride 1
Office of Naval Research (341)
800 No. Quincy Street
Arlington, VA 22217-5660

11. National Simulation Center (NSC) 1
ATTN: ATZL-NSC (Jerry Ham)
410 Kearney Avenue --- Building 45
Fort Leavenworth, KS 66027-1306

12. LT. Jim Patrey 1
Naval Air Warfare Center Training Systems Division
12350 Research Pkwy
Orlando, FL 32826-3224

13. LT. Karl R. Tenney 1
601 E. Glenwood Ave.
Fullerton, CA 92831-2714